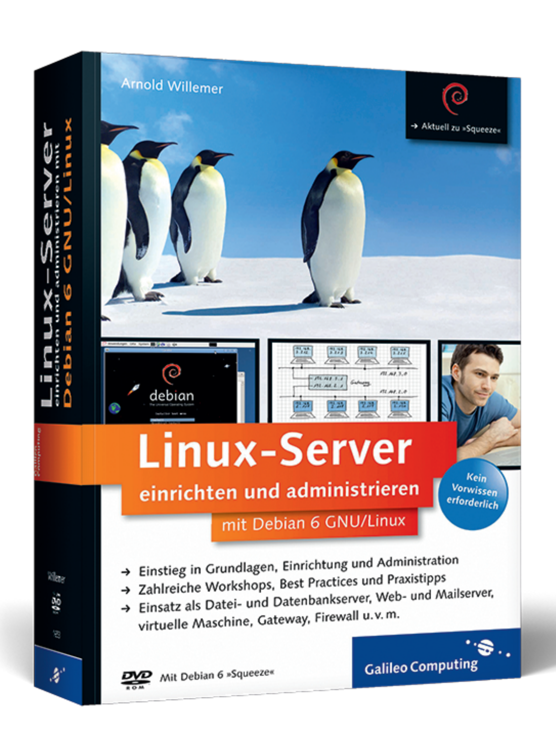
Arnold Willemer

Linux-Server einrichten und administrieren mit Debian 6 GNU/Linux





Auf einen Blick

Inhalt

Bevor Sie sich mit allen möglichen Hintergründen, Befehlen und Workshops beschäftigen, ist es vielleicht besser, erst einmal Fakten zu schaffen und einen Debian-Server einzurichten.

1 Installation eines Debian-Servers

Der erste Schritt

Falls Sie noch nie ein Debian-System installiert haben, sollten Sie nicht in Panik geraten. Heutzutage ist die Installation eines Debian-Systems keine Zauberei. Wenn Sie fertig sind, haben Sie ein System zur Verfügung, mit dem Sie alles das, was im weiteren Buch beschrieben wird, nachvollziehen können.

1.1 Von 0 auf 100 zum Server

Beschreiben wir zunächst, was das Ziel dieser Installation ist.

Zielvorstellung

Es soll ein Standard-PC als Debian-Server eingerichtet werden. Verwenden Sie dazu ruhig einen älteren PC, in dem sich eine leere Festplatte befindet oder eine, auf der alle Daten gelöscht werden können. Da wir eine Installation über das Internet machen, ist ein Breitbandzugang erforderlich. Neben dem Basissystem erhält der Server eine grafische Oberfläche. Datenbanken, Webserver und andere Dienste werden bei Bedarf nachinstalliert und bei der Grundinstallation nicht angewählt.

Reiner Debian-Server

Sie können neben Debian auch noch andere Betriebssysteme auf einer Festplatte installieren, die Sie beim Booten auswählen und dann starten. Es ist auch möglich, ein bereits installiertes Windows-System zu erhalten, in der Größe zu reduzieren und zusätzlich ein Debian-System zu installieren. Dies alles ist allerdings bei einem Server wenig sinnvoll, schließlich läuft ein Server meist durchgängig und wird nicht immer wieder neu gestartet.

Einfacher PC

Für Ihre ersten Experimente sollten Sie einen Standard-PC verwenden, gern ein wenig älter, aber bitte nicht so alt, dass Ihre jüngeren Kollegen

staunen, was einst einmal als Computer verkauft wurde. Je alltäglicher die Hardware ist, desto unproblematischer verläuft die Installation.

Virtueller Server

Sollten Sie keinen PC für Ihre Experimente zur Verfügung haben, können Sie für Ihre Experimente eine Virtualisierungssoftware wie VirtualBox[1] einsetzen.

1.1.1 Installationsmedium

Buch-DVD

Mit diesem Buch wird eine DVD mitgeliefert, die für die Installation eines Debian-Servers geeignet ist. Wenn Sie diese verwenden wollen, können Sie den Rest des Abschnitts überblättern und gleich in Abschnitt 1.1.2 ab Seite 32 weiterlesen. Die beigefügte DVD ist bootfähig und enthält die aktuelle Version Debian Squeeze. Sollten Sie mehr Pakete benötigen, als auf der DVD vorhanden sind, werden diese automatisch aus dem Internet nachgeladen.

Komplett-DVDs

Sie können einen kompletten Satz Debian-DVDs verwenden, um Debian zu installieren. Damit sind Sie bei der Installation unabhängig von einer Breitbandanbindung des Zielrechners. Solche DVDs können Sie bei diversen Händlern kaufen. Eine Übersicht der Anbieter finden Sie auf der Website von Debian *http://www.debian.de*.

Nachladen aus dem Internet

Wenn Sie eine schnelle Internetverbindung haben, können Sie diese auch herunterladen, wie Sie gleich sehen werden. Da aber nicht jeder die Software von acht DVDs auf seinen PC laden will, ist es auch möglich, nur mit einem Teil zu arbeiten. Die DVDs sind so organisiert, dass auf der ersten DVD die meistverwendete Software zu finden ist. Seltener benötigte Software erscheint auf einer späteren DVD. Bei der Installation erhalten Sie die Gelegenheit, dem System alle Medien bekannt zu machen. Installieren Sie später eine Software nach, erkennt Debian, ob sie auf einer der Medien ist, und fordert sie gegebenenfalls an. Alle anderen Softwarepakete beschafft das System aus dem Repository im Internet.

Medien aus dem Internet beschaffen

Debian im Internet

Sie können die Installationsmedien auch direkt aus dem Internet herunterladen. Unter der URL *http://www.debian.de* können Sie zwischen dem kompletten DVD-Satz oder der schlanken Netinstall-CD auswählen. Diese enthält nur das Nötigste, was zum Start der Installation erforderlich ist. Alle Installationsdaten kommen frisch aus dem Internet. Die Website von

1 VirtualBox siehe in Abschnitt 23.2 ab Seite 717 und *http://www.virtualbox.org*

Debian ist aber auch grundsätzlich ein guter Startpunkt auf der Suche nach Informationen, Dokumentationen und Foren.

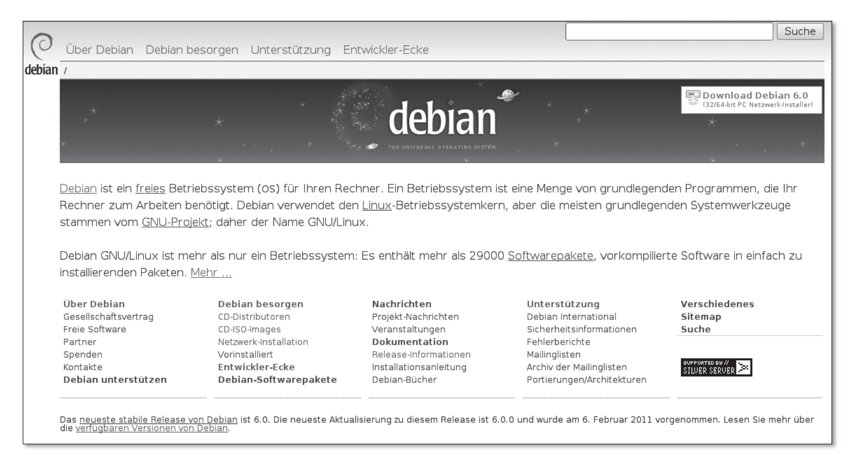


Abbildung 1.1 Website von Debian

Über den Punkt DEBIAN BESORGEN gelangen Sie auf die Website zum Download der Installationsmedien.

Images

Netinstall

Sofern Sie also keine Installations-DVD besitzen oder verwenden wollen, laden Sie hier eine Netinstall-CD herunter. Diese CD ist recht klein und längst nicht vollständig gefüllt. Sie dient nur als »Sprungbrett«, um für den Computer eine Internetverbindung aufzubauen, über die er dann seine eigentlichen Pakete installiert.

CD-ISO-Images

Unter den Links der Hauptseite (siehe Abbildung 1.1) finden Sie unter dem Titel DEBIAN BESORGEN den Stichpunkt NETZWERK-INSTALLATION. Diesen wählen Sie an. Auf der Folgeseite finden Sie Links mit verschiedenen Architekturen unter dem Titel KLEINE CDS. Sie befinden sich nun auf der Website *http://www.debian.de/distrib/netinst*.

Architektur

Architektur

Für die »Stable«-Version finden Sie eine Auswahl an unterstützten Computer-Architekturen. Diese Architekturen werden vor allem durch den Prozessor bestimmt, um den herum sie aufgebaut sind. Beispielsweise finden Sie einen Sparc-Prozessor in den UNIX-Maschinen von Sun und den PowerPC in den AIX-Maschinen von IBM.

```
amd64 armel kfreebsd-i386 kfreebsd-amd64 i386 ia64
mips mipsel powerpc sparc
```

32 oder 64 Bit

Bei einem PC funktioniert die Architektur »i386« in jedem Fall. Hat Ihr PC 64 Bit, können Sie auch »amd64« verwenden. Heutzutage werden nur noch 64-Bit-Rechner verkauft. Ob Sie einen Intel- oder AMD-Prozessor eingebaut haben, ist bei der Entscheidung unerheblich. Intel hatte die ersten 32-Bit-Prozessoren, und AMD war schneller bei 64 Bit. So ergaben sich die Namen. Der Hauptunterschied liegt in der Größe des Hauptspeichers. Ein 32-Bit-Prozessor kann architekturbedingt maximal 4 GB verwalten. Sollten Sie mittelfristig mehr Speicher benötigen, sollten Sie die 64-Bit-Version verwenden. Die Geschwindigkeitsvorteile einer 64-Bit-CPU sind nur bei Programmen spürbar, die im großen Umfang Fließkommaberechnungen durchführen. Bei manchen Anbietern von Fremdsoftware kann es passieren, dass nur eine 32-Bit-Version verfügbar ist.

Image-Datei

Klicken Sie die gewünschte Architektur mit der rechten Maustaste an, und speichern Sie die dahinterliegende Image-Datei auf Ihrem Rechner. Ein Image ist eine Datei, die die Oberfläche eines Datenträgers enthält. Die meisten Brennprogramme sind in der Lage, aus diesen Images eine CD oder DVD zu brennen. Ich werde darauf noch zurückkommen.

Alternative Download-Verfahren

Sie können das Image direkt herunterladen. Dazu werden die Protokolle FTP[2] (File Transfer Protocol) oder HTTP (Hypertext Transfer Protocol) verwendet. Allerdings sind vor allem die vollständigen DVD-Images derart groß, dass diese Techniken nicht besonders ideal sind. Der Browser kann nämlich eine einmal unterbrochene Übertragung nicht an derselben Stelle wieder aufsetzen. Bei einer Verbindungsstörung müssen Sie die Datei komplett neu übertragen. Das kostet unnötig Zeit, und der Debian-Server wird sinnlos zusätzlich belastet.

Jigsaw Download

Nur Neuheiten nachladen

Statt FTP und HTTP bietet Debian den Download per Jigsaw Download an, der kürzer als »jigdo« bezeichnet wird. Die Grundidee liegt darin, dass sich auf den verschiedenen Installationsmedien viele Dateien ständig wiederholen. Warum sollte also jede Datei mehrfach auf dem Server herumliegen? Stattdessen sammelt der jigdo-Manager die benötigten Dateien des Images vom Server ein, transportiert sie zum Anwender und erzeugt dort aus den Dateien ein Image. Besonders interessant wird der

2 FTP siehe Abschnitt 18.4 ab Seite 571

Jigsaw Download, wenn Sie bereits ein Image besitzen und es nur aktualisieren möchten. In diesem Fall werden nur die Dateien ausgetauscht, die sich verändert haben.

jigdo

Für die Verwaltung dieser Dateien gibt es das Programm `jigdo`. Wenn Sie bereits ein laufendes Debian- oder Ubuntu-System zur Verfügung haben, können Sie jigdo mit dem folgenden Befehl installieren:

```
debian # apt-get install jigdo-file
```

Für alle anderen Systeme finden Sie Unterstützung auf der folgenden Website:

http://www.atterer.net/jigdo

jigdo-Datei

Nun wählen Sie auf der Debian-Website die jigdo-Dateien aus, die für Ihre Architektur passen und laden sie herunter. Für jedes Image gibt es zwei Dateien. Eine endet auf *jigdo*, die andere auf *template*. Nehmen wir an, Sie wollten die erste DVD der nächsten Testversion von Debian für 64-Bit-Systeme herunterladen. Dann enden die passenden Dateien auf *amd64-DVD-1.jigdo* und *amd64-DVD-1.template*.

Sie starten nun das Programm mit dem Aufruf:

```
debian $ jigdo-lite debian-6.x-amd64-DVD-1.jigdo
```

Serverwahl

Bei der Rückfrage FILES TO SCAN geben Sie einfach die Return-Taste ein. Das Programm fragt nun nach dem Mirror-Server. Wenn Sie `jigdo` auf einem Ubuntu-Rechner starten, kann es sein, dass das Programm sogar einen Ubuntu-Server vorschlägt. Stattdessen geben Sie allerdings Folgendes ein:

http://ftp.de.debian.org/debian

Lokale Server

Wenn Sie nicht in Deutschland sind, verwenden Sie statt des »de« das Kürzel Ihres Landes. Das Programm sucht automatisch einen Server in Ihrer Nähe. Das Programm lädt nun eigenständig die benötigten Pakete und setzt sie zu einem Image zusammen. Wenn alles heruntergeladen ist, wird das Ergebnis gegen die Prüfsumme getestet, um zu gewährleisten, dass keine Lesefehler aufgetreten sind.

Verteilter Download per Bittorrent

Entlastung der Server

Das Bittorrent-Verfahren ist besonders geeignet, große Dateien zu verteilen. Das Prinzip beruht darauf, dass jeder Client auch gleichzeitig zum Server für die bereits heruntergeladenen Teile der Datei wird. Damit die Verteilung möglichst optimal wird, wird die Datei quasi streifenweise

zerteilt, und die Streifen werden in zufälliger Reihenfolge geladen. Es entsteht aus allen Interessierten einer Datei ein Verbund, der sich gegenseitig Teile der Datei anbietet. Damit wird der Server entlastet, und die Übertragungsgeschwindigkeit steigt.

Unterbrechbar

Aber auch für Teilnehmer mit geringer Bandbreite ist das Verfahren von Vorteil. So kann der Download jederzeit unterbrochen und später wieder aufgenommen werden. Es ist sogar möglich, die Download-Rate während des Tages so weit zu drosseln, dass der parallele Download beim Surfen nicht stört, und nachts die volle Bandbreite anzufordern.

Software

Statt der ISO-Datei wird eine Torrent-Datei heruntergeladen, die von dem Bittorrent-Programm verwendet wird. Bei den gängigen Linux-Varianten ist ein Bittorrent-Programm standardmäßig an Bord. Sie finden aber einen Bittorrent-Client für jede Plattform unter folgender URL:

http://www.bittorrent.com

Image-Datei brennen

Sollten Sie die Installation mit einer VirtualBox[3] ausführen wollen, können Sie die Image-Datei direkt als virtuelles CD-Laufwerk angeben. Sie müssen dann kein Medium brennen.

Brennerei

Der erste Teil wäre nun geschafft. Sie besitzen nun das Image einer CD als Datei. Nun müssen Sie aus der Image-Datei eine CD brennen. Falls Sie bereits ein Linux-System verwenden, klicken Sie die Image-Datei mit der rechten Maustaste an. Dort wird zum Brennen das Programm »Brasero« oder »k3b« angeboten, je nachdem, ob Sie GNOME oder KDE als Desktop einsetzen. Auch unter Windows führt ein Rechtsklick mit der Maus zum Ziel. Im Menü erscheint oben DATENTRÄGERABBILD BRENNEN, das Sie zu einem Dialog führt, der die Image-Datei auf einen CD-Rohling bannen kann. Auf dem Macintosh klicken Sie die Image-Datei an und wählen im Menü ABLAGE den Punkt DEBIAN...ISO AUF CD/DVD BRENNEN.

1.1.2 Booten der Installations-CD

Bootreihenfolge

Für die Installation müssen Sie von der Installations-CD booten. Bei manchen Computern reicht es, die CD beim Start einzulegen. In vielen Fällen wird der Rechner allerdings vorzugsweise von der Festplatte booten. Die Suche nach einer bootfähigen CD verzögert den Rechnerstart, doch sie wird ja nur bei der Installation von Systemen, also eher selten, benötigt.

3 VirtualBox siehe Seite 717

Bootmenü

Bei vielen Rechnern taucht der Hinweis auf, dass Sie ein Bootmedium auswählen können. Häufig muss dazu direkt nach dem Einschalten die Taste F12 oder F10 verwendet werden.

BIOS-Einstellung

Sollten Sie damit keinen Erfolg haben, können Sie im BIOS die Reihenfolge der Bootmedien so ändern, dass die CD bevorzugt wird. Auf den meisten Rechnern gelangen Sie über die Tasten F2 oder Entf ins BIOS. Da es verschiedene BIOS-Hersteller gibt, müssen Sie selbst nach dem Stichwort »Bootreihenfolge« suchen. Die Tasten zur Bedienung sowie kurze Hilfstexte finden Sie entweder unten oder am rechten Rand des Bildschirms.

US-Tastenbelegung

Berücksichtigen Sie, dass Sie im BIOS noch keinen deutschen Tastaturtreiber geladen haben. Die Tastenbelegung entspricht der amerikanischen Tastatur. Dort sind neben einigen Sonderzeichen auch die Tasten Z und Y vertauscht. Werden Sie also nach Y oder N gefragt, müssen Sie für die positive Antwort vermutlich die Taste Z drücken.

1.1.3 Die Installation beginnt

Wenn das Installationsmedium korrekt gestartet wurde, sollten Sie diesen Begrüßungsbildschirm vor sich sehen.

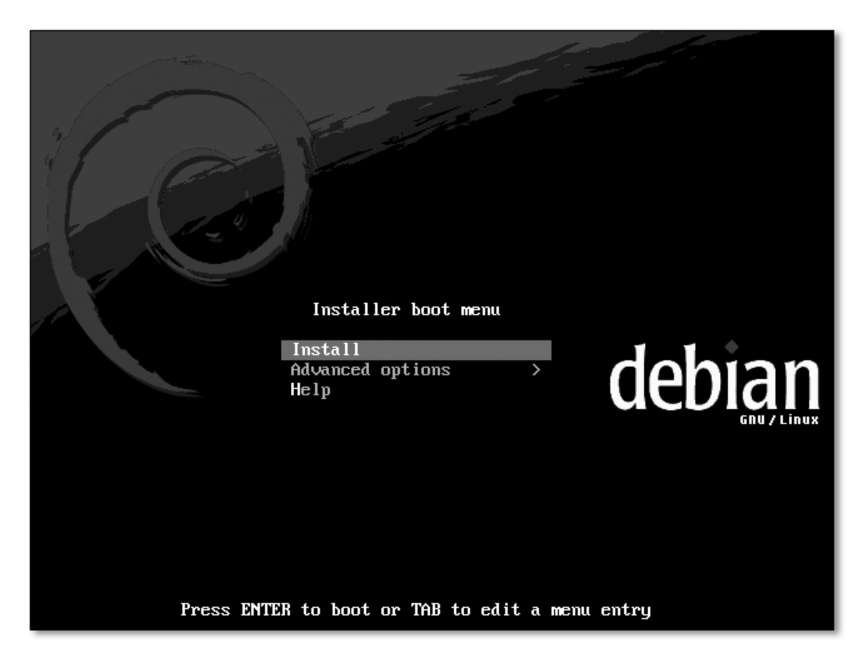


Abbildung 1.2 Installationsbeginn

Tastatur

Die Bedienung der Menüs ist sehr einheitlich. Mit der Return-Taste wird der Dialog bestätigt. Die Leertaste aktiviert einen Schalter, wenn sich der Cursor gerade darauf befindet. Die Elemente können mit den Pfeiltasten für oben und unten gewählt werden. Wenn in besonderen Situationen, wie beispielsweise in einer virtuellen Maschine, keine Cursor-Tasten zur Verfügung stehen, können auch die Tastenkombinationen Strg+P für oben und Strg+N für unten verwendet werden. Die Tabulatortaste wechselt zwischen mehreren Auswahlfeldern oder Eingabemöglichkeiten.

Wahl der grafischen Oberfläche

Auswahl

Vor dem eigentlichen Start der Installation können Sie die später verwendete grafische Oberfläche auswählen. Falls Sie also eine andere Oberfläche als GNOME verwenden möchten, ist es das Einfachste, diese vor dem eigentlichen Start der Installation auszuwählen. Dazu rufen Sie ADVANCED OPTIONS auf. Hier finden Sie den Punkt ALTERNATIVE DESKTOP ENVIRONMENTS. An dieser Stelle können Sie zwischen GNOME KDE, LXDE oder Xfce wählen.

GUI nicht notwendig ...

Wenn Sie Debian als Server einsetzen wollen, benötigen Sie keine grafische Oberfläche. Sie können alles von der Konsole aus administrieren. Auch eine Fernwartung über `ssh` stellt kein Problem dar.

... aber hilfreich

Auf der anderen Seite sind die Ressourcen für eine grafische Oberfläche nicht mehr so knapp, dass es notwendig wäre, darauf zu verzichten. Einige Bereiche wie die Druckeradministration oder Benutzereinrichtung ist per Mausklick schneller, einfacher und auch übersichtlicher zu machen. Vor allem können Sie bei einer grafischen Oberfläche mehrere Terminalfenster nebeneinander öffnen. Während Sie in dem einen Fenster Ihre Einstellungen machen, können Sie auf dem anderen Fenster die Protokolldateien beobachten.

GNOME

GNOME ist die Standardoberfläche bei Debian. Sie ist sehr einfach zu handhaben und bietet einige Administrationswerkzeuge, unter anderem Synaptic. Es ermöglicht die Softwarepaket-Installation auf sehr übersichtliche Weise. Sie finden dieses Programm im Menü SYSTEM • SYSTEMVERWALTUNG. Weitere Informationen dazu finden Sie in Abschnitt 1.2.1 ab Seite 44.

KDE

KDE ist die Alternative im Desktopbereich. Die Unterschiede zu GNOME aufzuführen ist etwa so sinnvoll wie die Beschreibung der Unterschiede zweier Bundesligavereine. Beide Oberflächen haben ihre Fans und werden keine Argumente für die andere gelten lassen. Sie können KDE aber

auch als Alternative nachinstallieren und bei jedem Einloggen entscheiden, wem Sie heute den Vorzug geben.

```
apt-get install kde
```

Sie werden dabei allerdings feststellen, dass KDE komplett auf Englisch eingestellt ist. Um dies zu ändern, installieren Sie die deutsche Lokalisierung.

```
apt-get install kde-l10n-de
```

LXDE

LXDE ist ein ressourcensparender Desktop, der ideal sein dürfte, wenn die grafische Oberfläche vor allem dazu dienen soll, mehrere Terminalsitzungen in Fenstern nebeneinanderstellen zu können.

Xfce

Auch Xfce gilt als ressourcenschonender Desktop. Er kann vor allem GNOME und KDE-Programme sehr gut integrieren.

Experten-beschreibung

Dieser Abschnitt beschreibt die Installation eines Debian-Systems anhand einer Experten-Installation. Sollten Sie lieber die normale Installation verwenden, wird auch sie in dieser Form ablaufen. Allerdings wird Debian Sie nicht so oft nach Ihrer persönlichen Meinung fragen. Charakterstarke Menschen können das ertragen. Beim Lesen sollten Sie sich einfach nicht wundern, wenn Ihnen bei Ihrer Installation nicht jede Frage begegnet.

Region

Regionales

Mit der Return-Taste werden die Fragedialoge gestartet, die erkunden, wie Ihr zukünftiges Debian-System aussehen soll. Als Erstes wird die Sprache abgefragt. Da Sie dieses Buch offensichtlich gut lesen können, wird Deutsch keine schlechte Wahl sein. Die Amerikaner haben mit der Aussprache von »Deutsch« so ihre Schwierigkeiten. Also finden Sie hier den Begriff »German«. Im nächsten Schritt wird der Standort geklärt. Hier haben Sie eine reiche Auswahl von Belgien bis zur Schweiz. Die Tastaturbelegung kann unabhängig von Sprache und Standort gewählt werden.

Netzwerkumgebung

Automatik mit DHCP

Im nächsten Schritt wird die Netzwerkhardware untersucht und in der Regel auch auf Anhieb erkannt. Anschließend wird das Netzwerk eingerichtet. Sofern im Netzwerk ein DHCP-Server[4] läuft, der Neuzugängen einen Weg ins Internet weist, können Sie diesem getrost die Netzwerkeinrichtung überlassen. Selbst wenn Sie bislang nicht wussten, dass Sie

4 DHCP siehe Abschnitt 7.7 Seite 273

so etwas besitzen, kann es sein, dass ein solcher Server existiert, wenn Sie beispielsweise einen DSL-Router verwenden, um ins Internet zu gelangen.

Manuelle Einrichtung

Sollten Sie keinen DHCP-Server haben, benötigen Sie die IP-Adresse des Rechners, des Gateways und des DNS-Servers. Selbst wenn Sie einen DHCP-Server im Netz haben, gibt es gute Gründe, auf dessen Unterstützung zu verzichten. Schließlich soll ein Server nicht jeden Tag eine andere IP-Adresse haben. Bei der automatischen Installation wird aber bei Vorhandensein eines DHCP-Servers keine manuelle Einstellung angeboten. Um sie zu erzwingen, können Sie beim Start der Installation das Ethernetkabel herausziehen. So wird die DHCP-Konfiguration verlässlich scheitern. Natürlich müssen Sie den Stecker nach Eingabe der Adressen dann wieder einstecken.

Beispielkonfiguration

Bei einer manuellen Installation geben Sie die Netzwerkparameter an. Für die IP-Adresse verwenden Sie beispielsweise 192.168.1.12. Die Netzmaske setzen Sie beispielsweise auf 255.255.255.0. Das Standard-Gateway und der DNS-Server könnten beispielsweise beide mit 192.168.1.1 angegeben werden, wenn dies die Adresse des Routers ist. Abschließend werden Sie noch einmal gefragt, ob alles korrekt eingegeben wurde.

Hostname

Sie werden gefragt, wie der Name des Rechners lauten soll. Dieser Name wird Hostname genannt; er unterscheidet ihn von den anderen Computern in Ihrem Netzwerk. Geben Sie dem Rechner einen markanten Namen. Im nächsten Dialog werden Sie nach dem Domainnamen gefragt. Dies ist etwas vereinfacht der Name des Netzwerks, in dem sich Ihr Server tummelt. Im Buch verwende ich dafür *willemer.edu*. Benutzen Sie einen eigenen Namen Ihrer Wahl. Er sollte möglichst nicht bereits im Internet vorkommen, sofern Sie nicht genau wissen, was Sie tun. Wenn Sie Ihre Domäne beispielsweise *google.de* nennen, wird es Ihnen schwerfallen, die gleichnamige Suchmaschine im Internet zu erreichen.

1.1.4 Die Festplatte partitionieren

Manuell oder geführt?

Nun steht die Partitionierung der Festplatte an. Hier werden Optionen mit LVM (Logical Volume Manager) angeboten, die an anderer Stelle in diesem Buch behandelt werden.[5] Ansonsten steht die Möglichkeit, die Partitionierung von Hand durchzuführen oder die vollständige Festplatte

5 Zum Thema LVM finden Sie in Abschnitt 12.3.4 ab Seite 378 weitere Informationen.

zu verwenden und sich dabei der Führung von Debian anzuvertrauen. Sie stoßen dabei auf folgende Auswahl:

- Geführt – verwende vollständige Festplatte
- Geführt – gesamte Platte verwenden und LVM einrichten
- Geführt – gesamte Platte mit verschlüsseltem LVM
- Manuell

Geführt

Gesamte Festplatte geführt

Sie dürfen die gewünschte Festplatte auswählen. In den meisten Computern ist nur eine eingebaut. So entstehen immerhin keine nagenden Zweifel, welche gemeint sein könnte. Nun werden drei Möglichkeiten angeboten. Sie können eine große Partition verwenden. Dies ist für Anfänger empfehlenswert und wird auch von uns gewählt. Wenn Sie die Partitionierung anders einrichten wollen, finden Sie in Abschnitt 12.3 ab Seite 372 eine ausführlichere Erläuterung zum Thema. Die Installationsroutine richtet zwei Partitionen ein: eine große Datenpartition für System- und Nutzerdateien und eine Swap-Partition. Nach einer weiteren Rückfrage werden die Partitionen angelegt, und das Grundsystem wird eingerichtet.

Manuell

Wählen Sie statt der geführten Partitionierung den Punkt Manuell, landen Sie in einem Dialog, in dem Sie neben einigen weiteren Menüpunkten eine Liste der Festplatten und deren Partitionen finden.

Neue Partitionstabelle

Wenn Sie die komplette Festplatte auswählen, erhalten Sie die Möglichkeit, eine neue Partitionstabelle anzulegen und damit alle vorhandenen Partitionen zu löschen.

Freier Speicher

In der Partitionstabelle kann ein Bereich als freier Speicher angegeben sein. Wenn Sie diesen anwählen, erhalten Sie die Möglichkeit, eine neue Partition anzulegen. Sie geben an, welche Größe die Partition haben soll. Der Dialog zeigt Ihnen daraufhin die maximal zur Verfügung stehende Größe an. Sie werden gefragt, ob es sich um eine primäre oder logische Partition handeln soll. Solange Sie nicht mehr als vier Partitionen benötigen, können Sie alle als primäre Partitionen einrichten.

Typ und Einhängepunkt

Debian bemerkt, dass Sie nicht den gesamten freien Platz verwenden und fragt, ob Sie die Partition am Anfang oder am Ende der Platte ausrichten wollen. Nachdem Sie hier Anfang angegeben haben, erreichen Sie einen

Dialog, in dem Sie angeben müssen, welchen Dateisystemtyp Sie auf dieser Partition nutzen wollen. Für ein Debiansystem empfehlen sich die Dateisystemtypen »ext3« und »ext4«. Hier ist »ext3« das altbewährte System und »ext4« dessen Nachfolger. Dann müssen Sie den Einhängepunkt angeben, und hier sollten Sie einen einsamen Schrägstrich verwenden. Über den Punkt ANLEGEN DER PARTITION BEENDEN schreiben Sie diese Partition in die Partitionstabelle.

Partition bearbeiten

Wenn Sie eine vorhandene Partition anwählen, können Sie diese »benutzen«. Das bedeutet, dass Sie den Typ des Dateisystems angeben und die Stelle, an der die Partition im Verzeichnisbaum eingehängt wird. Beispielsweise könnten Sie auf der Partition ein ext3-Dateisystem anlegen und diese als Wurzelverzeichnis (/) verwenden. Darüber hinaus geben Sie an, ob die Partition neu formatiert werden soll. Dies sollten Sie nur dann unterlassen, wenn Sie gute Gründe haben, warum Sie die Dateien auf der Partition noch benötigen. Sie können die Partition nachträglich in ihrer Größe verändern oder aber die Partition komplett löschen.

Swap

Für die Swap-Partition wiederholen Sie das Anlegen der Partition mit leichten Variationen. Wieder wechseln Sie auf FREIER SPEICHER. Sie erstellen wieder eine neue Partition. Diesmal können Sie bei der Frage nach der Größe alles nehmen. Wieder wird es eine primäre Partition.

Im Folgedialog wählen Sie als Benutzung AUSLAGERUNGSSPEICHER (SWAP). Einen Einhängepunkt brauchen Sie hier nicht anzugeben. Sie wählen ANLEGEN DER PARTITION BEENDEN.

Abschluss

Sie schließen die Partitionierung ab und übernehmen die Änderungen. Sie müssen ein weiteres Mal bestätigen, dass die Änderungen auf die Festplatte geschrieben werden sollen. Nun, da die Zielpartition fertig ist, werden die auf dem Installationsmedium vorhandenen Dateien auf die Festplatte kopiert und installiert.

1.1.5 Benutzer einrichten

Passwort root

Ein Benutzer existiert auf jedem Debian-System, und das ist der Administrator, der immer den Benutzernamen root trägt. Für diesen wird ein Passwort benötigt. Da root alles darf, sollte das Passwort nicht allzu trivial sein. Nach der Eingabe wird das Passwort ein zweites Mal angefordert, damit Tippfehler ausgeschlossen werden.

Einfach merken, schwer zu knacken

Ein gutes Passwort sollte nicht in einem Lexikon stehen, kein Name sein und nicht zu wenig Buchstaben enthalten. Vielfach wird sogar empfohlen, dazwischen eine Zahl oder ein Sonderzeichen einzufügen. Und natürlich

dürfen Sie es nicht aufschreiben. Schon gar nicht auf einen dieser gelben Klebezettel, die anschließend am Monitorrand befestigt werden. Das tut nämlich jeder! Ein kleiner Kniff hilft hier. Denken Sie sich einen Satz aus, der Ihnen Freude bereitet und den Sie sich leicht merken können. Beispielsweise: »Ich bin ein toller Hecht und der Stolz der ganzen Firma«. Nun nehmen Sie die Anfangsbuchstaben: IbetHudSdgF. Das sieht schon ganz toll aus. Wenn Sie nun an einem 24. Geburtstag haben, fügen Sie an der vierten Stelle eine 2 ein: Ibe2tHudSdgF. Auf dieses Passwort kommt kein Mensch, und Sie freuen sich jedes Mal, wenn Sie es eintippen.

Normalo

Das Administrationskonto darf niemals für normale Aktivitäten verwendet werden. Dazu hat der Administrator immer ein zusätzliches, »ziviles« Konto. Für dieses fordert das Installationsprogramm anschließend die Informationen an. Es beginnt mit dem vollen Namen des Benutzers. Anschließend wird aus dem Vornamen ein Benutzername gebildet, den Sie allerdings ändern können. Der Benutzername ist der, mit dem Sie sich am System anmelden. Abschließend wird auch für diesen Benutzer ein Passwort eingegeben und durch erneute Eingabe bestätigt.

1.1.6 Pakete installieren

Repository-Server

Auch wenn Sie von den vollständigen Installationsmedien installieren, sollten Sie einen Debian-Paket-Server für Ihren Computer einstellen. Hier bekommen Sie Updates und Upgrades her. Das ist vor allem im Falle von Sicherheitslücken wichtig. Die Debian-Server weisen sich gegenüber Ihrem System mit Zertifikaten aus. Dadurch wird gewährleistet, dass Ihnen nicht schädliche Software untergeschoben wird. Von diesen Servern erhalten Sie auch die Softwarepakete, die Sie später nachinstallieren. Solange Sie Ihre Software dort beziehen, riskieren Sie nicht, dass Ihnen Fremde böse Software unterschieben.

Regionale Server

Damit die Pakete für den Downlaod nicht sinnlos um die halbe Welt segeln, sollen Sie das Land eingeben, aus dem die Pakete heruntergeladen werden sollen. Anschließend erscheint eine Liste der Spiegelserver aus diesem Land. Wählen Sie einen beliebigen Server aus.

Proxy

Die Pakete werden über das Protokoll HTTP geladen. Falls der zu installierende Rechner nur über einen Proxy[6] ins Internet kommt, müssen Sie nun die Proxydaten eingeben. Sie erfahren Sie vom Netzwerkadministrator. Sofern Ihr Netzwerkadministrator nichts vorschreibt, lassen Sie das Feld am besten leer.

6 Proxy siehe Abschnitt 10.3 Seite 339

Nachfragen

Es werden Ihnen nun ein paar Fragen gestellt, bei denen Sie einfach die Vorgaben belassen können. Eine Rückfrage betrifft die Dienste, die nun gestartet werden sollen. Es ist »cron« aufgeführt. Mit WEITER kommen Sie zum nächsten Schritt. Sie werden jetzt aufgefordert zu bestätigen, dass die Laufwerk-Geräte-IDs umbenannt werden. Sie werden gefragt, ob Sie an der Paketverwendungserfassung teilnehmen würden. Damit versucht das Debian-Team zu ermitteln, welche Pakete oft installiert werden. Diese Informationen beeinflussen die Verteilung der Pakete auf die DVDs.

Schließlich werden Sie gefragt, welche Software installiert werden soll. Sie bekommen folgende Liste angeboten:

- GRAFISCHE DESKTOP-UMGEBUNG
- WEB-SERVER
- DRUCK-SERVER
- DNS-SERVER
- DATEI-SERVER
- MAIL-SERVER
- SQL-DATENBANK
- SSH-SERVER
- LAPTOP
- STANDARD-SYSTEMWERKZEUGE

Hauptpakete wählen

Vorgewählt ist der erste und der letzte Punkt. Wenn Sie eine grafische Oberfläche wollen, lassen Sie den Punkt angewählt. Wenn Sie ihn abwählen, ist die Installation deutlich schneller. Sie müssen dann allerdings den Server komplett über die Tastatur warten.

Entscheidungen

Die Entscheidungen sind nicht endgültig. Sie können alle Pakete auch nachträglich installieren. Durch die spätere Auswahl können Sie die Pakete sehr viel genauer auswählen. Für den Anfang würde ich Ihnen durchaus zu einer grafischen Desktopumgebung raten. Auf die Standard-Systemwerkzeuge sollten Sie keinesfalls verzichten. Wenn Sie ein Notebook verwenden, sollten Sie auch das Laptop-Paket wählen, und ein SSH-Server sollte eingerichtet werden, wenn Sie Ihren Server über das Netzwerk warten wollen.

Bootloader

Nach einiger Zeit des Ladens und Installierens wird GRUB, der Bootloader, installiert. Prinzipiell kann er sowohl in der Partition als auch im

MBR (Master Boot Record) installiert werden. Um das System starten zu können, muss GRUB in den MBR. Die Anfragen sind etwas verwirrend, da man zuerst fortfahren muss, ohne den Bootloader zu installieren, um ihn dann doch im MBR einrichten zu können. Das Thema Bootloader wird in Abschnitt 12.9 ab Seite 408 behandelt.

Abschluss

Nun ist die Installation beendet, und das System startet neu. Sie können sich nun anmelden, den Rechner erkunden und den Beispielen im Buch folgen. Wenn Sie den Rechner wieder ausschalten wollen, melden Sie sich als root an und fahren dann das System mit dem Befehl `halt` herunter. Sollten Sie eine grafische Oberfläche installiert haben, finden Sie im Menü SYSTEM den Menüpunkt zum Herunterfahren des Systems.

1.2 Softwarepakete nachinstallieren

Nachdem ein Debian-Server installiert wurde, ist er bereits mit der Grundausstattung an Software versorgt. Wenn Sie weitere Software benötigen, können Sie diese über die gleichen Server nachinstallieren, von denen Sie auch Ihr Debian-System bekommen haben. Auch wenn es Updates gibt, werden sie über diese Server geliefert. Debian hat im Laufe der Jahre ein ausgeklügeltes System geschaffen, um die Installation von Software stabil und sicher zu gestalten. Die herausragenden Fähigkeiten dieses Systems umfassen vor allem folgende Punkte:

- Der Anwender kann sich mit entsprechenden Werkzeugen einen Überblick über die Software verschaffen und nach Programmen suchen, die seine Probleme lösen.
- Die Softwarepakete können aus verschiedenen Quellen geladen werden. Es können sowohl beliebig viele Server im Internet als auch CDs, DVDs oder lokale Repositorys genutzt werden. Dies alles ist für den Anwender unsichtbar. Er erhält einfach die Software, die zusammenpasst und auf dem aktuellsten Stand ist.
- Es ist möglich, einen kompletten Release-Wechsel ohne Eingriff des Anwenders herbeizuführen.
- Die Programmpakete stammen aus gesicherter Quelle. Es wird nur dann von einem Server Software bezogen, wenn er sich mit seiner Signatur ausweist.

Abhängigkeiten

Die Installation von Programmen ist nicht ganz so trivial, wie es dem Anwender scheint. Beispielsweise wird das Programm Kaffeine, mit dem

man prima fernsehen kann, in einer GNOME-Umgebung nicht so richtig glücklich, da es für den Desktop KDE geschrieben wurde und dessen Bibliotheken benötigt. Wenn Sie nun die Version von Kaffeine auftreiben, die unter KDE 3.5 vorbildlich lief und einfach zu bedienen war, so würde sie auf einem aktuellen KDE 4.x nicht laufen, weil sich die Bibliotheken inzwischen verändert haben.

Rekursive Installation

So enthält das Programmpaket für Kaffeine das Programm selbst sowie die wichtigsten Konfigurationsdateien und einen Verweis auf die KDE-Bibliotheken, die es zum Laufen benötigt. Diese Verweise enthalten auch die benötigte Versionsnummer der Bibliothek. Soll das Kaffeine-Paket installiert werden, wird geprüft, ob die passenden Bibliotheken bereits installiert sind. Sind diese nicht vorhanden, werden auch sie nachgeladen und installiert. Auch die Pakete, die von den Bibliotheken benötigt werden, werden wiederum automatisch in die Installation integriert, sodass es passieren kann, dass das Einrichten eines einzigen Programms eine ganze Reihe von Paketen nach sich zieht. All dies leistet das Debian-Installationssystem in vorbildlicher Weise.

Top Down

Das Debian-Paket-System ist komplex und daher nicht ganz einfach zu verstehen. Damit Sie möglichst schnell einen Einstieg in den Umgang mit Debian-Paketen gewinnen, beginne ich mit der Beschreibung der grafischen Werkzeuge. Diese ermöglichen eine schnelle Übersicht und macht das Erfassen leichter. Durch die Netzwerkfähigkeit des X-Protokolls ist es möglich, die Anwendungen fernzusteuern, wie es in Abschnitt 1.2.1 gezeigt wird. Als Administrator ist es aber auch wichtig, mit den zugrunde liegenden Programmen auf der Konsole umgehen zu können. In manchen Fällen müssen Sie über eine einfache ssh-Verbindung fernwarten. Wenn Sie dann nur einen Windows-Rechner zur Verfügung haben, nützt Ihnen die Netzwerkfähigkeit von X nicht viel, da Windows sie nicht beherrscht.

1.2.1 Grafisch installieren

Die grafische Oberfläche GNOME stellt mehrere Werkzeuge zur Verfügung, um mit der Debian-Paketverwaltung arbeiten zu können. KDE bietet eigene Werkzeuge mit Adept. Aber Sie können Synaptic oder das Software-Center auch aufrufen, wenn Sie KDE als Desktop verwenden.

Software-Center

Anwendergerecht

Die einfachste Methode, an Software zu gelangen, stellt das Software-Center dar. Dieses erreichen Sie über das Menü System • Systemverwaltung • Software-Center. Hier ist die Software so angeordnet, dass ein

durchschnittlicher Anwender die Software finden wird, die er für seine Anwendungen benötigt. Die Programme sind nach Themen geordnet. Der Anwender klickt sich durch die Kategorien, bis er das Programm gefunden hat, das er installieren will.



Abbildung 1.3 GNOME Software-Center

Homebanking-Beispiel

Wenn der Anwender beispielsweise ein Programm für das Homebanking sucht, kann er die Sparte BÜRO anklicken und findet dort das Programm HOMEBANK. Für jedes Programm gibt es eine eigene Seite, die einen Bildschirmabzug und eine kurze Beschreibung enthält, wie in Abbildung 1.4 zu sehen.

Installation

Wenn der Anwender nun den Button INSTALLIEREN – KOSTENLOS anklickt, wird er nach dem root-Passwort gefragt. Das ist erforderlich, da Softwareinstallation zu den Aufgaben des Administrators gehört. Ohne dessen Befugnis darf Software nicht installiert, gelöscht oder geändert werden. Das dient als Schutz, damit keine Schadsoftware eingerichtet werden kann.

Anschließend wird das Programm heruntergeladen, eingerichtet und steht sofort unter dem Menü ANWENDUNGEN • BÜRO • HOMEBANK zur Verfügung.

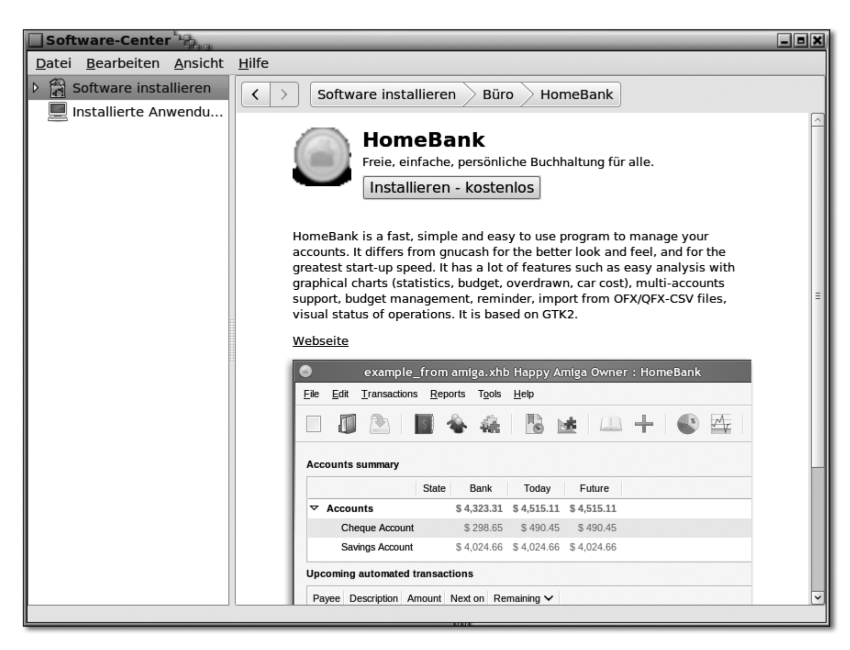


Abbildung 1.4 Software-Center Programmbeschreibung

Synaptic

Während sich das Software-Center eher an den reinen Anwender richtet, ist Synaptic mehr das Allround-Tool mit vielen Möglichkeiten. Das Programm Synaptic finden Sie über das Menü SYSTEM • SYSTEMVERWALTUNG • SYNAPTIC-PAKETVERWALTUNG. Anschließend werden Sie nach dem Administratorpasswort gefragt.

Fernwartung

Auch wenn Sie an einem anderen PC sitzen, können Sie Synaptic aufrufen. Voraussetzung ist allerdings, dass Sie einen Linux-Computer verwenden, der ein X Window System installiert hat. In diesem Fall rufen Sie den Zielrechner, der im Beispiel *debian* heißt, per `ssh`[7] mit der Option `-X` für eine Umleitung des Displays auf.

```
linux $ ssh -X debian
arnold@debian's password:

Last login: Fri Aug 27 03:40:33 2010 from hape.local
arnold@debian:~$
```

root-Rechte

Dort rufen Sie nun das Programm `synaptic` auf. Das Fenster wird auf dem Bildschirm Ihres Arbeitsplatzrechners erscheinen, da der Standard-Display ja über die SSH-Verbindung umgeleitet wird. Da Synaptic aller-

7 siehe Abschnitt 9.3 Seite 310

dings root-Rechte benötigt, rufen Sie es über das Programm `gksu` auf, das Sie zunächst nach dem root-Passwort des Rechners debian fragen wird.

```
arnold@debian:~$ gksu synaptic
```

GUI hier, Aktion da

Das Programm erscheint nun auf Ihrem Arbeitsplatz und lässt sich genau so bedienen, als würde es hier laufen. Tatsächlich arbeitet aber nur die grafische Oberfläche hier. Alle Aktionen werden auf dem Rechner debian ausgeführt.

Hauptfenster

Das Programm Synaptic hat ein gegliedertes Hauptfenster, wie in Abbildung 1.5 zu sehen ist. Rechts oben steht eine Liste der Softwarepakete. In dem Feld darunter findet sich eine Beschreibung des angewählten Pakets. Links oben befindet sich eine Reihe von Auswahlfeldern, mit denen beispielsweise bestimmte Themen selektiert werden können. Der Inhalt dieser Liste wird durch die Buttons bestimmt, die nach Sektionen, dem Status und anderen Filtern auswählen können.

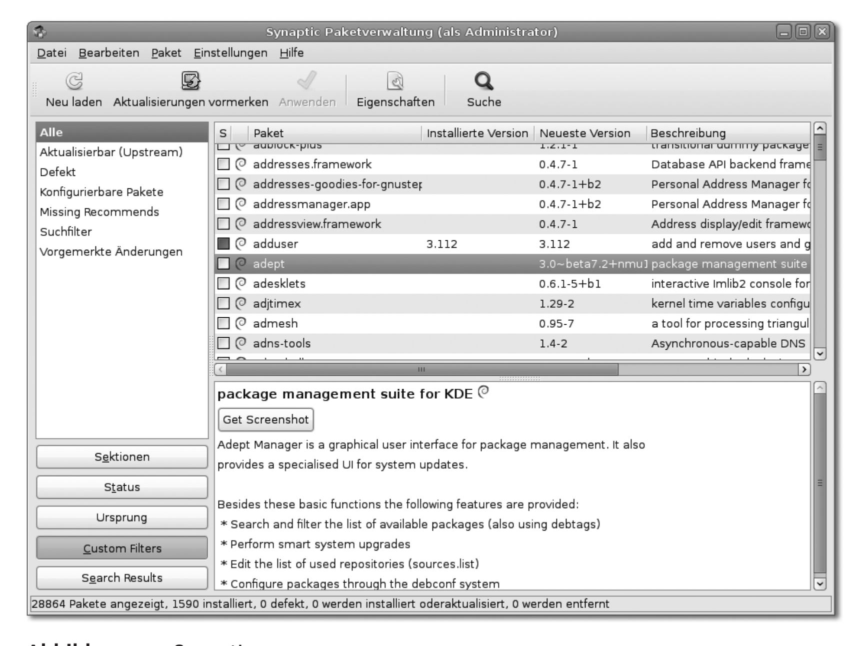


Abbildung 1.5 Synaptic

Software suchen

In der oberen Leiste befindet sich der Button SUCHE. Wie der Name bereits erahnen lässt, können Sie damit einen Dialog starten, in dem Sie Stichworte eingeben, die das Paket spezifizieren. Nach der Eingabe reduziert sich die Liste auf diejenigen Pakete, die alle angegebenen Stichwörter im Namen oder in der Beschreibung enthalten.

Zur Installation anwählen

Wenn in der Liste ein Paket erscheint, das Ihren Vorstellungen entspricht, können Sie es anklicken. Im Fensterbereich rechts unten erscheint eine Beschreibung. Wollen Sie das Paket installieren, klicken Sie den Eintrag in der Liste mit der rechten Maustaste an oder klicken in das Kästchen links neben dem Eintrag. Es erscheint ein Menü, das den Punkt ZUM INSTALLIEREN VORMERKEN enthält. Sobald Sie diesen angewählt haben, erscheint gegebenfalls ein Dialog, der Sie informiert, falls weitere Pakete vorgemerkt werden müssen, um das ausgewählte Paket zu installieren. Wenn Sie diesen Dialog bestätigen, erhält das Kästchen einen kleinen Pfeil, und der Button ANWENDEN in der Hauptleiste wird aktivierbar. Sie können nun weitersuchen und weitere Pakete zur Installation vormerken oder beispielsweise installierte Pakete löschen.

Entfernen

Pakete, die ein grünes Kästchen besitzen, sind bereits installiert. Wenn Sie dieses Kästchen anklicken, sind in dem daraufhin erscheinenden Menü Punkte zum Entfernen des Pakets freigeschaltet. Sie können allerdings auch das Paket erneut installieren.

Änderungen anwenden

Sobald Sie den Button ANWENDEN anklicken, erscheint ein Dialog, der Sie über den Umfang der gewünschten Installationen oder Deinstallationen aufklärt. Sobald Sie diesen mit ANWENDEN bestätigen, startet die eigentliche Installation. Ein Verlaufsbalken zeigt den Download. Dann wird die Konfiguration gemeldet. In deren Verlauf kann es passieren, dass Sie nach bestimmten für die Installation erforderlichen Informationen gefragt werden. Anschließend erscheint wieder das Hauptfenster von Synaptic.

1.2.2 Grafisch aktualisieren

Aufruf

Die grafischen Oberflächen bieten auch ein Werkzeug, mit dem die Updates durchgeführt werden können. Unter GNOME findet man es unter dem Menüpunkt SYSTEM • SYSTEMVERWALTUNG • AKTUALISIERUNGSVERWALTUNG. In der Grundeinstellung startet es sogar automatisch und meldet die vorliegenden Paketaktualisierungen.

Kandidaten

In der oberen Liste werden alle Pakete angezeigt, für die Aktualisierungen vorliegen. Dabei sind diese untergliedert in sicherheitskritische Updates und solche, die Programmfehler beseitigen oder Erweiterungen mitbringen. Sie können auch einzelne Pakete von der Aktualisierung ausschließen. Dabei werden abhängige Pakete automatisch ausgeschlossen.

Prüfen

Mit dem Button PRÜFEN können Sie die Liste der Aktualisierungen noch einmal laden. Durch Klicken auf den Button AKTUALISIERUNGEN INSTALLIEREN startet die Installation.

Wenn Sie ein Paket mit der Maus anwählen, können Sie im unteren Fenster die Liste der Änderungen des Pakets sehen. Dies ermöglicht eine Abschätzung, wie eilig die Aktualisierung des Pakets ist.

Detailinformationen



Abbildung 1.6 Aktualisierungsverwaltung

1.2.3 Aufgabe per Tasksel wählen

Grobauswahl

Das Programm Tasksel kennen Sie bereits von der Installation. Es eignet sich vor allem dazu, mehrere Programmpakete auf einmal zu installieren und damit die Hauptaufgaben des Computers festzulegen.

Menü

Sie wählen damit vor, ob der Computer mit einer grafischen Oberfläche versehen ist oder ob es sich um einen Laptop handelt. Ansonsten können Sie verschiedene Serverarten vorgeben.

Rufe Aptitude

Wenn Sie den Punkt MANUELLE PAKETAUSWAHL auswählen, wird das Programm Aptitude aufgerufen.

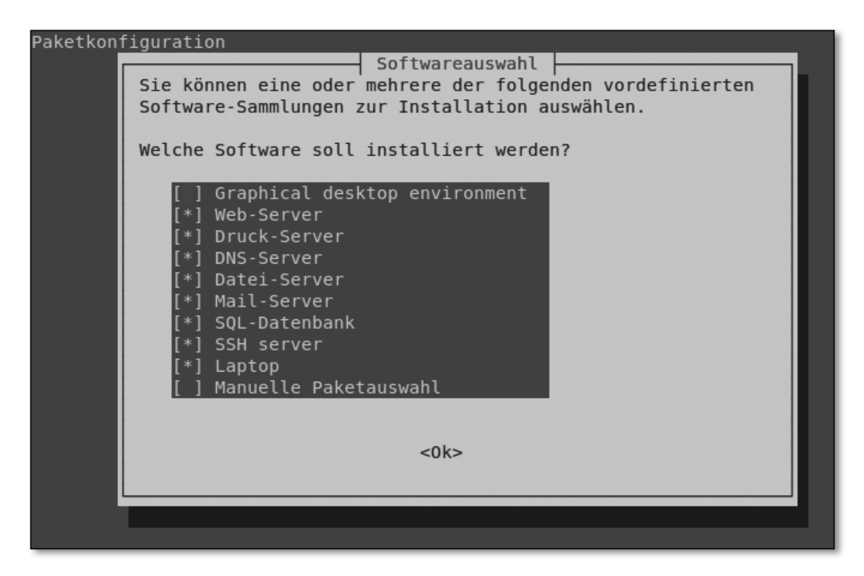


Abbildung 1.7 Tasksel

1.2.4 Aptitude installiert

Paketauswahl

Das Programm Aptitude ist eine Terminallösung für den Umgang mit den Softwarepaketen. Seine besondere Fähigkeit gegenüber dem Aufruf der anderen APT-Konsolenprogramme ist die erleichterte Auswahl von Softwarepaketen, wie es beispielsweise Synaptic beherrscht. Gegenüber Letzterem benötigt es keine grafische Umgebung und kann darum per SSH gestartet werden und auch noch auf sehr schlanken Umgebungen eingesetzt werden.

Bildschirm-aufteilung

Der Bildschirm ist zweigeteilt. In der oberen Liste sehen Sie die Kategorien für die Installationen. Sie können mit dem weiß hinterlegten Balken über eine Kategorie fahren und sie mit der Return-Taste auswählen. Darunter finden Sie weitere Untergliederungen, bis Sie auf die einzelnen Pakete stoßen. Im unteren Bereich des Bildschirms werden die Pakete, Kategorien und Aktivitäten näher beschrieben.

Menü

In der obersten Zeile finden Sie ein Menü, das Sie über die Tastenkombination [Strg]+[T] erreichen. Die wichtigsten Tastenkürzel werden allerdings bereits in der zweiten Zeile angezeigt. Mit dem Kommando `q` können Sie das Programm verlassen.

```
 Aktionen  Rückgängig  Paket  Auflöser  Suchen  Optionen  Ansichten  Hilfe
C-T: Menü ?: Hilfe q: Beenden u: Update g: Download/Inst./Entf. von Paketen
aptitude 0.6.3       #Kaputt: 3   Werde 4.862kB mehr belegen     DL-Größe: 1.211
--- Aktualisierbare Pakete (342)
--- Neue Pakete (2416)
--- Installierte Pakete (1197)
--- Nicht installierte Pakete (24896)
--- Veraltete und selbst erstellte Pakete (13)
--- Virtuelle Pakete (3655)
--- Tasks (657)

Eine neuere Version dieser Pakete ist verfügbar.

Diese Gruppe enthält 342 Pakete.

[1(1)/...] Schlage 3 Beibehaltungen vor
e: Prüfen  !: Anwenden  .: Nächste  ,: Vorhergehende
```

Abbildung 1.8 Aptitude

Aktualisierung

Das Kommando `u` sorgt dafür, die Paketverzeichnisse zu aktualisieren, und entspricht dem Befehl `apt-get update`. Sie können dies auch über das Menü AKTIONEN • PAKETLISTE AKTUALISIEREN erreichen.

Suchen

Mit dem Schrägstrich können Sie per Stichwort nach Programmpaketen suchen. Bei der Eingabe jeder Taste wird die Liste der Pakete kleiner.

Paketauswahl markieren

Die Pakete werden mit bestimmten Kommandos dahingehend markiert, ob sie installiert +, deinstalliert - oder komplett gelöscht _ werden sollen. Dazu wandern Sie mithilfe der Pfeiltasten und der Return-Taste durch die Kategorien und verwenden die in der Tabelle 1.1 aufgeführten Kommandos, um die Pakete zu markieren. Weitere Befehle finden Sie unter dem Menü PAKET. Auf der rechten Seite jedes Menüpunkts finden Sie auch ein Tastenkürzel für die jeweilige Funktion.

Kommando	Wirkung
+ (Plus)	Paket installieren
- (Minus)	Paket deinstallieren
_ (Unterstrich)	Paket komplett deinstallieren

Tabelle 1.1 Markierungskommandos für Pakete

Markierungen ausführen

Wenn Sie mit der Auswahl fertig sind, rufen Sie über das Menü [Strg] + [T] den Punkt AKTIONEN • INSTALLIEREN/ENTFERNEN VON PAKETEN auf. Sie können auch das Kommando `g` verwenden. Achten Sie darauf, dass Sie das kleine »g« verwenden und dass Sie sich dabei in der Hauptebe-

ne befinden. Daraufhin wird das Programm kurzzeitig verlassen, und es erscheinen die Meldungen der `apt-get`-Aufrufe, die das Programm generiert hat. Das Programm bittet um die Eingabe der Return-Taste und kehrt dann wieder zu `aptitude` zurück.

1.2.5 Auf apt-get getippt

Einzelbefehle

Der Befehl `apt-get` eignet sich ideal dazu, einzelne Kommandos abzusetzen, um Programmpakete zu installieren, zu entfernen oder Updates anzustoßen. Direkt auf das Kommando `apt-get` folgt ein Befehl, der angibt, welche Aktion benötigt wird.

Installation: apt-get install

apt-get install

In diesem Buch werden Sie am häufigsten auf die Befehlskombination `apt-get install` stoßen, wenn darauf hingewiesen wird, welche Pakete für welche Aufgaben benötigt werden. Durch den Aufruf von `apt-get` mit dem Parameter `install` und dem Paketnamen wird die Installation ausgelöst. Der Befehl zeigt an, welche Pakete aufgrund von Abhängigkeiten hinzuinstalliert werden müssen. Es wird gemeldet, wie viele Daten heruntergeladen werden müssen und welchen Platz das zu installierende Paket auf der Festplatte einnehmen wird. Anschließend bekommen Sie noch einmal die Möglichkeit, die Installation abzubrechen oder zu bestätigen.

Aufruf von apt-get install
`apt-get install` <*Paketname*> [<*Paketname*>] *

Die Installation des Apache-Servers gelingt mit dem Aufruf

```
debian # apt-get install apache2
```

`apt-get` gibt die eigentliche Installation an `dpkg` weiter. Falls dieser aber Softwarepakete vermisst, sucht `apt-get` zunächst die Pakete zusammen und lässt dann `dpkg` die Pakete in der richtigen Reihenfolge installieren.

Pakete entfernen

Das Programm `apt-get` kann auch für die Deinstallation eines Softwarepakets sorgen. Dabei prüft es, welche anderen Pakete von dem zu entsorgenden Paket betroffen sind, und schlägt diese ebenfalls zur Deinstallation vor.

apt-get remove

Die Entfernung kennt allerdings drei Intensitäten. Der Befehl `remove` entfernt die Softwaredateien von der Festplatte, belässt aber die Konfiguration. Auf diese Weise bleiben die bisherigen Einstellungen erhalten, sollte die Software später wieder installiert werden.

Aufruf von apt-get remove

```
apt-get remove <Paketname> [ <Paketname> ] *
```

apt-get purge

Soll auch die bisherige Konfiguration entfernt werden, weil nicht beabsichtigt ist, das Paket noch einmal zu installieren, oder weil Sie eben bewusst die bisherige Konfiguration loswerden möchten, dann ist der Befehl `purge` der richtige Befehl.

Aufruf von apt-get purge

```
apt-get purge <Paketname> [ <Paketname> ] *
```

In beiden Fällen liegt allerdings das Softwarepaket, aus dem die Software installiert wurde, noch auf dem Rechner. Es ist jedoch nicht entpackt. Wenn Sie das Paket also erneut installieren, wird kein neuer Download veranlasst, weil aus dem lokalen Archiv-Cache installiert werden kann. Im Verzeichnis */var/cache/apt/archives* findet sich noch die Debian-Paket-Datei.

apt-get clean

Soll mit dem Softwarepaket auch der Archiv-Cache aufgeräumt werden, so verwenden Sie den Befehl `clean`.

Aufruf von apt-get clean

```
apt-get clean <Paketname> [ <Paketname> ] *
```

apt-get autoremove

Sollten Pakete liegen bleiben, die nicht mehr benötigt werden, erkennt das System auch dieses. Bei den Installationsaufrufen werden Sie darauf aufmerksam gemacht, dass Sie sie mit dem Befehl `apt-get autoremove` bereinigen können. Der Befehl benötigt keine Parameter.

Aufruf von apt-get autoremove

```
apt-get autoremove
```

Bestätigung

Bevor das große Aufräumen stattfindet, werden Sie allerdings noch einmal auf den Umfang der Arbeiten hingewiesen und erhalten eine Chance, alles wieder abzublasen.

```
debian # apt-get autoremove
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut
Status-Informationen einlesen... Fertig
Die folgenden Pakete werden ENTFERNT:
  dvgrab frei0r-plugins gnome-audio gnome-pilot gnome-pi...
  libasound2-plugins libavahi-core6 libboost-regex1.42.0...
  ...
0 aktualisiert, 0 neu installiert, 54 zu entfernen
und 338 nicht aktualisiert.
Nach dieser Operation werden 79,8MB Plattenplatz freigegeben.
Möchten Sie fortfahren [J/n]?
```

Paketlistenverzeichnis aktualisieren

apt-get update

Die Informationen, welche Softwarepakete zur Verfügung stehen, werden lokal gehalten. Allerdings kann es unter bestimmten Bedingungen passieren, dass diese Informationen nicht mehr synchron mit dem Stand im Internet sind. Sie merken dies beispielsweise daran, dass Sie Fehlermeldungen bekommen, die Datei sei nicht verfügbar. In diesem Fall sollten Sie sich ein neues Verzeichnis aus dem Internet holen. Der Befehl dazu lautet `apt-get update`. Da sich der Befehl auf das gesamte Verzeichnis bezieht, benötigt er keine Parameter.

Aufruf von apt-get update
`apt-get update`

[!]

Der Begriff »Update« bezieht sich hier also nicht etwa auf das Update der Programmpakete, sondern auf die Aktualisierung der Paketinformationen.

1.2.6 Software aktualisieren

apt-get upgrade

Um die installierte Software auf den aktuellen Stand zu bringen, brauchen Sie nur den Befehl `apt-get upgrade` einzugeben.

Aufruf von apt-get upgrade
`apt-get upgrade`

Aktualisierung

Das Programm ermittelt daraufhin die verfügbaren Aktualisierungen aller installierten Softwarepakete. Vor dem Herunterladen werden die neueren Pakete aufgezählt, und ihr Umfang wird angezeigt. Sie haben dann die Wahl, ob Sie fortfahren möchten. Anschließend werden die Pakete heruntergeladen und installiert.

```
debian # apt-get upgrade
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut
Status-Informationen einlesen... Fertig
Die folgenden Pakete sind zurückgehalten worden:
  bind9 bind9-host bind9utils cpp-4.4 dnsutils festival
  ...
  xserver-xorg-video-chips xserver-xorg-video-sis
293 aktualisiert, 0 neu installiert, 0 zu entfernen
und 60 nicht aktualisiert.
Es müssen 439MB an Archiven heruntergeladen werden.
Nach dieser Operation werden 1.189kB Plattenplatz
zusätzlich benutzt.
Möchten Sie fortfahren [J/n]?
Hole:1 http://ftp...queeze/main base-files 5.9 [68,5kB]
Hole:2 http://ftp.../main perl-modules 5.10.1-14 [3.481kB]
...
0% [2 perl-modules 2.691kB/3.481kB 77%] ... 2S 31Min 35s
```

apt-get dist-upgrade

Der Befehl `dist-upgrade` geht deutlich weiter als der normale `upgrade`. Letzterer wird nicht weiterarbeiten, wenn ein Konflikt auftritt. Dagegen wird `dist-upgrade` weniger wichtige Pakete deinstallieren, wenn es damit möglich ist, ein wichtiges Paket zu installieren.

```
debian # apt-get dist-upgrade
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut
Status-Informationen einlesen... Fertig
Die folgenden Pakete sind zurückgehalten worden:
  bind9 bind9-host bind9utils cpp-4.4 dnsutils festival ...
  ...
Die folgenden Pakete werden aktualisiert:
  apache2 apache2-mpm-prefork apache2-utils apache2.2-bin...
  ...
  xserver-xorg-video-chips xserver-xorg-video-sis ...
278 aktualisiert, 0 neu installiert, 0 zu entfernen
und 60 nicht aktualisiert.
Es müssen 437MB an Archiven heruntergeladen werden.
Nach dieser Operation werden 1.250kB Plattenplatz zusätzlich
benutzt.
```

```
Möchten Sie fortfahren [J/n]?
...
```

1.2.7 Paketquellen anpassen

sources.list

Es stellt sich natürlich die Frage, woher `apt-get` denn weiß, von wo es die Pakete beziehen soll. Dazu gibt es die Konfigurationsdatei */etc/apt/sources.list*. Darin stehen die Paketquellen. Das können CDs, aber auch URLs aus dem Internet sein. Um die Pflege dieser Datei müssen Sie sich zunächst nicht weiter kümmern.

Struktur sources.list

Die Datei *sources.list* ist eine normale Textdatei. Sie können sie mit Ihrem Lieblingseditor bearbeiten. Und wie immer im Leben ist es auch hier gut, wenn Sie wissen, was Sie tun. So finden sich in meiner Konfiguration unter anderem folgende Zeilen:

```
deb http://ftp2.de.debian.org/debian/ squeeze main
deb-src http://ftp2.de.debian.org/debian/ squeeze main
deb http://security.debian.org/ squeeze/updates main
```

Die Spalten sind durch Leerzeichen getrennt und enthalten folgende Informationen:

- **Pakettyp**
  Hier steht typischerweise »deb« oder »deb-src«, je nachdem, ob es sich um ein binäres, also ausführbares Paket oder um die Quelltexte handelt.
- **URI**
  In der zweiten Spalte wird angegeben, wo die Daten zu finden sind. Drei Arten von Adressen werden Sie hier normalerweise vorfinden. Mit »cdrom« werden CDs oder DVDs bezeichnet. »http« ist für Server, die die Daten nach dem HTTP-Protokoll[8] verbreiten, und »ftp« für Server, die das FTP-Protokoll[9] verwenden. Wenn Sie ein eigenes, lokales Repository aufbauen wollen, können Sie auch »file« als Adresse verwenden.
- **Distribution**
  Hier steht in der aktuellen Version squeeze oder squeeze/updates.
- **Komponenten**
  Hier können Attribute wie main, non-free, contrib oder andere Begrif-

8 siehe Abschnitt 20.4 Seite 632
9 siehe Abschnitt 18.4 Seite 571

fe stehen, die die verfügbaren Pakete einschränken. Auch stable und unstable können hier aufgeführt werden.

apt-cdrom add

Bei der Installation wird das Installationsmedium in die Quellen eingetragen. Eine Ausnahme bilden lediglich die Netinstall-Medien, deren Software gerade ausreicht, um die Installation zu starten, und dann alle Pakete aus dem Internet zieht. Bei der Installation geben Sie an, in welchem Land Sie wohnen. Daraus wird ermittelt, welche Server für Sie am günstigsten sind. Auch die werden zu den Installationsquellen hinzugefügt. Weitere CDs oder DVDs können Sie jederzeit mit dem Befehl `apt-cdrom add` zu den Installationsquellen hinzufügen.

1.2.8 Debian-Paket-Manager

Installationsknecht

Die Basisanwendung für die Installation und Deinstallation von Debian-Paketen heißt `dpkg`. Letztlich ist es dieses Programm, das von Synaptic bis `apt-get` aufgerufen wird. Das Programm ist in der Lage, ein Paket zu entpacken und alle Bestandteile an die richtige Position zu bringen. Es ist später auch in der Lage, das Paket wieder zu deinstallieren. Das Programm weiß ja, wo die Dateien hingekommen sind, und sammelt sie alle wieder auf.

Abhängigkeiten

Das Programm `dpkg` kann auch die im Softwarepaket hinterlegten Abhängigkeiten auslesen und wird melden, wenn versucht wird, ein Paket zu installieren, obwohl die Voraussetzungen dafür nicht vorliegen. Andersherum wird es eine Deinstallation verweigern, wenn das Paket noch von anderen Paketen benötigt wird.

Begrenzte Fähigkeiten

`dpkg` ist nicht in der Lage, diese Abhängigkeiten aufzulösen. Es kann nur lokal vorliegende Paketdateien verarbeiten und nicht eigenständig Dateien aus dem Repository nachladen. Damit liegt die Versionsverwaltung in den höheren Werkzeugen wie etwa `apt-get`.

Aufruf von dpkg

```
dpkg -i <DebianPaket>
dpkg -r <DebianPaket>
dpkg -P <DebianPaket>
```

Optionen

Die Option `-i` bewirkt die Installation des Pakets. Die Option `-r` entfernt die Installation, abgesehen von den Konfigurationsdateien. Bei erneuter Installation wird so die bisherige Konfiguration beibehalten. Ist das nicht

gewünscht,kann mit der Option -P oder --purge auch das Löschen der Konfigurationsdateien veranlasst werden.

Paketsuche

Wenn Sie nach bestimmten Paketen suchen, können Sie mit der Option -l nach bestimmten Worten suchen. Wenn Sie einen Stern benutzen, kann es sinnvoll sein, das Suchmuster in Anführungszeichen zu setzen.

```
debian # dpkg -l *ldap*
Status=Nicht/Installiert/Config/U=Entpackt/halb konFiguriert/
         Halb installiert/Trigger erWartet/Trigger anhängig
/ Fehler?=(kein)/R=Neuinstallation notwendig (Status, Fehler:
         GROSS=schlecht)
||/ Name           Version        Beschreibung
+++-==============-==============-=======================...
un  ldap-client    <keine>        (keine Beschreibung vor...
un  ldap-server    <keine>        (keine Beschreibung vor...
ii  ldap-utils     2.4.17-2.1     OpenLDAP utilities
ii  libaprutil1-ld 1.3.9+dfsg-3   The Apache Portable Run...
un  libldap-2.3-0  <keine>        (keine Beschreibung vor...
ii  libldap-2.4-2  2.4.17-2.1     OpenLDAP libraries
un  libldap2       <keine>        (keine Beschreibung vor...
un  libnet-ldap-pe <keine>        (keine Beschreibung vor...
ii  libnss-ldap    264-2.2        NSS module for using LD...
un  libnss-ldapd   <keine>        (keine Beschreibung vor...
ii  libpam-ldap    184-8.5        Pluggable Authenticatio...
un  libsasl2-modul <keine>        (keine Beschreibung vor...
un  openldap-utils <keine>        (keine Beschreibung vor...
un  openldapd      <keine>        (keine Beschreibung vor...
un  proftpd-mod-ld <keine>        (keine Beschreibung vor...
un  smbldap-tools  <keine>        (keine Beschreibung vor...
un  sudo-ldap      <keine>        (keine Beschreibung vor...
un  umich-ldap-uti <keine>        (keine Beschreibung vor...
un  umich-ldapd    <keine>        (keine Beschreibung vor...
```

Paketinhalt

Sie können auch feststellen, welche Datei durch welches Paket installiert wurde. Dabei hilft Ihnen die Option -S Der folgende Aufruf ermittelt, dass das Programm testparm zum SAMBA-Paket gehört.

```
debian # dpkg -S  testparm
samba-doc: /usr/share/doc/samba-doc/...pages/testparm.1.html
samba-common-bin: /usr/share/man/man1/testparm.samba3.1.gz
samba-common-bin: /usr/bin/testparm.samba3
```

GNOME-Version

Wenn Sie die grafische Oberfläche GNOME verwenden, ist das Programm GDebi installiert, das die Funktionalität von dpkg enthält. Wenn eine Paketdatei auf dem Desktop liegt, können Sie sie mit der rechten Maustaste anklicken und aus dem Menü den Punkt Mit GDebi öffnen auswäh-

len. Das Programm GDebi wird ein paar Informationen über das Paket ausgeben und eine Installation anbieten.



Abbildung 1.9 GDebi: GNOME-GUI für dpkg

GDebi wird prüfen, ob die Paketabhängigkeiten erfüllt sind, und abbrechen, wenn dies nicht der Fall ist. GDebi wird die Abhängigkeiten nicht auflösen und keine weiteren Pakete hinzuziehen.

1.3 Source-Pakete manuell installieren

Bei einem Open-Source-Betriebssystem sollte man es nicht für ungewöhnlich halten, dass man auch mal ein Paket als Source bekommt und dieses installieren soll. Allerdings gehört dies heutzutage zu den Ausnahmen. Am ehesten kommt man in die Verlegenheit, wenn eine spezielle Hardware eingebunden werden muss, für die der Hersteller den Quellcode zur Verfügung stellt.

1.3.1 Vorarbeiten

Für alle Debian-Pakete können Sie den Quellcode erhalten. Das garantiert Ihnen die GPL (GNU Public Licence). Der Quellcode wird als Debian-Paket geliefert und beispielsweise mit `dpkg` ausgepackt.

Fremde Pakete

Wenn die Quelltexte nicht in einem Debian-Paket kommen, werden sie vermutlich als gepackte Tar-Datei geliefert. Diese können mit dem Bordmittel `tar`[10] ausgepackt werden. Um Speicherplatz und Transferzeit zu

10 siehe Abschnitt 15.5 ab Seite 464

sparen, ist das Paket zumeist komprimiert. Sie erkennen dies an der Endung *tgz*.

Werkzeuge installieren

Sie werden für das Installieren mindestens das Programm `make` benötigen, das Sie, sofern Sie es nicht bereits an Bord haben, durch das Paket gleichen Namens nachinstallieren können. Mit großer Wahrscheinlichkeit wird die Software in C oder C++ geschrieben sein. Zur Übersetzung benötigen Sie den GNU-Compiler. Das Paket heißt g++.

```
debian # apt-get install make g++
```

Verzeichnis anlegen

Für das Auspacken des Pakets sollten Sie ein Verzeichnis anlegen. Hat der Anbieter die Daten nicht in einem Verzeichnis gebündelt, müssen Sie ansonsten die einzelnen Dateien aus Ihren Dateien heraussuchen.

```
debian $ mkdir src
debian $ mv sourcepaket.tgz src
debian $ cd src
debian $ tar xvz sourcepaket.tgz
...
```

Voraussetzungen prüfen

Durch die Option `v` können Sie sehen, wie die Daten abgelegt sind. War der Anbieter ordentlich und hat noch ein Verzeichnis beigefügt, müssen Sie nun dort hineinwechseln. Hier befindet sich meist das Programm `configure`. Dieses prüft, ob die Umgebung vorhanden ist, um das Programm zu übersetzen und zu installieren. In manchen älteren Paketen wird der gleiche Effekt durch den Aufruf von `make depend` erreicht.

```
debian $ ./configure
```

Das Programm `configure` ist recht geschwätzig. Es zählt auf, was es alles für die Übersetzung benötigt, und stoppt, wenn etwas wirklich Wichtiges fehlt. Sie müssen dann die fehlenden Bestandteile nachinstallieren, bevor Sie noch einmal `configure` aufrufen. Sobald das Programm problemlos durchgelaufen ist, können Sie die Übersetzung mit dem Befehl `make` starten.

```
debian $ make
```

Übersetzung

Auch dieses Programm wird reichlich Ausgaben produzieren. Mit diesem Befehl starten Sie nämlich die Übersetzung des Pakets. Das kann zwischen ein paar Minuten bis zu mehreren Stunden in Anspruch nehmen, je nachdem, wie umfangreich die Software ist. Irgendwann sollte aber die Übersetzung zu einem erfolgreichen Ende kommen. Dann muss die Software installiert werden. Auch dafür wird das Programm `make` verwendet.

```
debian $ su -
debian # make install
```

Nachteile

Tatsächlich funktioniert diese Form der Einrichtung in vielen Fällen sehr gut. Allerdings gibt es zwei erhebliche Nachteile dieses Verfahrens. Erstens installieren Sie am Debian-Paket-Management vorbei. Automatische Updates und die Überprüfung auf Paketabhängigkeiten können mit Software, die auf diese Weise installiert wurde, nicht gelingen. Zweitens kann es sein, dass diese Übersetzung misslingt. Und sofern Sie nicht selbst programmieren können, müssen Sie es lernen[11], jemanden fragen, der sich damit auskennt, oder aufgeben.

1.3.2 make macht das schon

Abhängigkeiten

Das Programm `make` ist ein wichtiges Werkzeug für Programmierer, die damit die Übersetzung ihrer Programme organisieren. Wenn Sie das Programm ohne weitere Parameter starten, wird es im aktuellen Verzeichnis zunächst nach der Datei *Makefile* und dann nach der Datei *makefile* suchen. In dieser findet `make` die Anweisungen, was es »machen« soll. Im Makefile können Regeln aus Abhängigkeiten von Dateien definiert werden, die zu Aktionen führen. Insbesondere wenn eine Datei nicht existiert oder älter ist als eine andere, wird die Aktion aufgerufen. Dies ist vor allem für Programmierer wichtig. Wenn Sie etwas an einer Quelltextdatei geändert haben, ist sie neuer als das Programm. Darum muss die Quelltextdatei vom Compiler übersetzt werden, damit eine neue Version des Programms entsteht. Es müssen aber nicht alle Module übersetzt werden, sondern nur diejenigen, die geändert wurden. Am einfachsten ist ein Makefile anhand eines Beispiels zu verstehen.

[zB]

Stellen Sie sich vor, Sie seien Programmierer und Sie wollen das Programm *meinprog* erstellen. Dieses besteht aus den Source-Dateien *haupt.c*, *test.c* und *tools.c*. Jede dieser Dateien hat eine Header-Datei (*haupt.h*, *test.h* und *tools.h*), die die Datei jeweils selbst einbindet. Header-Dateien enthalten Code, der Deklarationen beinhaltet, von denen die Source-Dateien abhängen. Ändert sich eine Header-Datei, betrifft das jede Source-Datei, die sie einbindet. Dazu bindet *haupt.c* jede andere Header-Datei ein, und jedes Modul bindet die globalen Definitionen aus *haupt.h* ein. Wenn eine Source-Datei übersetzt wird, entsteht eine Objektdatei. Die aus der Übersetzung der Datei *haupt.c* entstehende Datei heißt *haupt.o*. Im letzten

11 Es gibt im Verlag Galileo Press das Buch »Einstieg in C++« von diesem außerordentlich sympathischen und kompetenten Autor, dessen Name mir gerade nicht einfällt.

Schritt werden die Objektdateien durch den Linker zusammengebunden, und es entsteht so die Datei *meinprog*.

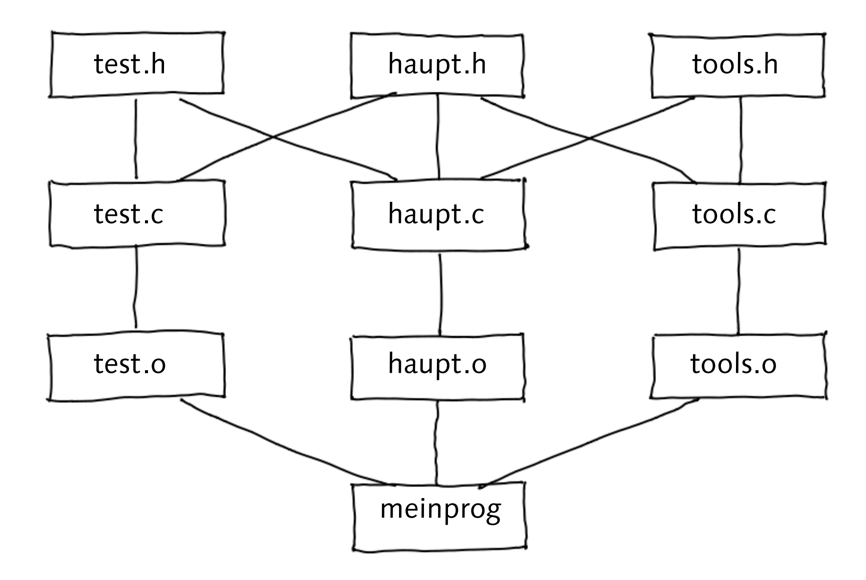


Abbildung 1.10 Beispielprojekt für make

Sie legen nun eine Datei namens *Makefile* an und beschreiben darin den Weg der Übersetzung vom Quelltext zum fertigen Programm. Das Programm *meinprog* hängt von den Dateien *test.o*, *haupt.o* und *tools.o* ab. Die Formulierung in einem Makefile ist folgendermaßen:

```
meinprog:  test.o haupt.o tools.o
```

Diese Zeile beschreibt die Abhängigkeit der Datei *meinprog* von den Dateien *test.o*, *haupt.o* und *tools.o*. Man bezeichnet *meinprog* als »Ziel« oder englisch als »target«. Die auf die Abhängigkeitsregel folgenden Zeilen beschreiben, wie die Zieldatei erstellt wird.

```
meinprog:  test.o haupt.o tools.o
        cc -o meinprog test.o haupt.o tools.o
```

Die Datei *meinprog* wird generiert, indem der Linker mit `cc -o` gestartet wird. Dabei werden die Objektdateien als Parameter übergeben. Solche Aktionszeilen müssen mit einem Tabulator beginnen. Es dürfen keine Leerzeichen verwendet werden.

Nun soll `make` noch wissen, wie die Objektdateien erzeugt werden. Die Abhängigkeiten bestehen einerseits zur jeweiligen Source-Datei, aber auch zu jeder Header-Datei, die die Source-Datei mit einbindet.

```
test.o : test.c test.h haupt.h

tools.o : tools.c tools.h haupt.h

haupt.o : haupt.c haupt.h test.h tools.h
```

Die Aktionszeile ruft den Compiler mit `cc -c` auf. Er erzeugt eine gleichnamige Objektdatei zu der angegebenen Source-Datei.

```
test.o : test.c test.h haupt.h
        cc -c test.c
```

Geringster Aufwand

Das Programm `make` ermittelt nun anhand seiner Regeln, wie man mit minimalem Aufwand Zieldateien aus den Quelldateien generieren kann. `make` erkennt, wenn eine der Quelldateien neuer als die Zieldatei ist, und ruft die Generierungsprogramme auf, bis die Zieldateien neuer als die jeweiligen Quellen sind oder eine Aktion scheitert.

Zusammenfassend lassen sich die Einträge in der Datei *Makefile* folgendermaßen darstellen:

Grundstruktur eines Makefile-Eintrags

<Ziel> : *<Abhängigkeiten>*
        <Generierungskommando>

Regeln

Diese Grundstruktur nennt man Regel. Eine neue Regel muss mit einer Leerzeile von der vorherigen getrennt werden. Denken Sie daran: Der Leerraum vor dem Generierungskommando muss ein Tabulatorzeichen sein. Es können auch mehrere Kommandozeilen nacheinander angegeben werden. Alle müssen mit einem Tabulator eingerückt sein. Die Kommandozeilen werden jeweils in einer separaten Shell abgearbeitet. In manchen Situationen erzeugt das Seiteneffekte, die Sie berücksichtigen müssen. Probieren Sie einmal das folgende Beispiel für eine *Makefile*-Datei aus:

```
try :
       cd .. ; pwd
       pwd
```

Die Ergebnisse der beiden Aufrufe von `pwd` sind nicht gleich. Der Wechsel mit `cd ..` gilt nur für die aktuelle Zeile. In der nächsten Zeile wird wieder im bisherigen Verzeichnis gearbeitet:

```
cd .. ; pwd
/home/arnold/my/src/unix
pwd
/home/arnold/my/src/unix/make
```

Hängen also Kommandos so zusammen, dass sie in einer gemeinsamen Shell bearbeitet werden müssen, sollten sie in dieselbe Zeile geschrieben und mit Semikolons getrennt werden. Bei langen Zeilen kann mit einem Backslash die Zeile in der nächsten Zeile fortgesetzt werden. Als Kommentarzeichen gilt # in der ersten Spalte.

Variablen im Makefile

make arbeitet mit Variablen

Durch die Verwendung von Variablen können die Makefiles besser strukturiert und flexibler gestaltet werden. Wie in einer Shell können Sie auch im Makefile einer Variablen eine Zeichenkette zuweisen und deren Inhalt durch Voranstellen eines $-Zeichens vor den Variablennamen auswerten.

[zB] Im Beispiel werden die Objektdateien zusammen behandelt und zweimal aufgezählt; einmal in der Abhängigkeitsbeschreibung von *meinprog* und dann im Compileraufruf:

```
meinprog:  test.o haupt.o tools.o
        cc -o meinprog test.o haupt.o tools.o
```

Hier können Sie eine Variable OBJS für die Auflistung der Objektdateien definieren. Durch Einsetzen von OBJS ergibt sich folgende Makedatei:

```
OBJS = test.o haupt.o tools.o

meinprog: $(OBJS)
      cc -o meinprog $(OBJS)
```

Die Variablen müssen nicht im Makefile selbst definiert werden. `make` kann auf Umgebungsvariablen zurückgreifen, die von der aufrufenden Shell festgelegt wurden.

Vordefinierte Variablen

Es ist möglich, mehrere Ziele mit einer Regel zu behandeln. So könnte beispielsweise $(OBJS) als Ziel verwendet werden. Die einzelnen Ziele werden nacheinander aufgelöst. Im Generierungskommando kann auf das aktuelle Ziel Bezug genommen werden. Dazu gibt es vordefinierte Variablen, die in Tabelle 1.2 am Beispiel *haupt.o* aufgezeigt werden.

Variablen	Bedeutung
$@	Dateiname des Ziels (haupt.o)
$*	Basisname des Ziels (haupt)

Tabelle 1.2 Vordefinierte make-Variablen

Suffixregeln

Die Suffixregeln beschreiben den Übergang einer Dateiendung zu einer anderen. Eine solche Regel erkennen Sie daran, dass als Ziel die zwei Dateiendungen mit dem jeweiligen Punkt am Anfang direkt hintereinander stehen:

```
.quell.ziel:
```

Der typischste Übergang ist sicher der von C-Sourcen zu Objekten. Die Sourcen enden auf .c und die Objektdateien auf .o. Die entsprechende Suffixregel lautet dann:

```
.c.o:
        cc -c $<
```

Die interne Variable `$<` darf nur bei Suffixregeln verwendet werden und bezeichnet das aktuelle Ziel.

Mehrere Ziele

Voneinander abhängige Ziele

Ein Makefile kann mehrere Programme generieren. Diese Fähigkeit wird dann eingesetzt, wenn gleiche Quelltexte für mehrere Projekte gebraucht werden, die vielleicht sogar noch voneinander abhängig sind. Ein typisches Beispiel sind Client- und Serverprogramme, die in den Headern gleiche Datenstrukturen verwenden:

```
all: client server

client: $(SENDHEADER) $(COMMONOBJS) $(CLTOBJS)
        ...

server: $(SENDHEADER) $(COMMONOBJS) $(SRVOBJS)
        ...
```

Zielabhängigkeiten

Das erste Ziel ist immer das Ziel des gesamten Makefiles. In diesem Fall würde also beim Aufruf von `make` erst das Ziel »all« generiert werden. Da es keinerlei verbundene Aktion gibt, wird lediglich geprüft, ob die Abhängigkeiten erfüllt sind. Entsprechend wird als Nächstes das Ziel »client« und dann das Ziel »server« gebildet. Es ist nicht zwingend, aber üblich,

das Pseudoziel, das alle Programme eines Makefiles generiert, »all« zu nennen.

make als Installationstool

Wenn Makefiles zur Installation verwendet werden, wird ein Pseudoziel »install« eingeführt, das überprüft, ob alle Dateien des Projekts an den richtigen Stellen vorhanden sind, und ansonsten als Aktion einfache Kopierbefehle absetzt. Sie können das Ziel »install« direkt als Parameter von `make` aufrufen:

```
make install
```

»Das ist meine Mupfel!«
Der Pinguin Ping in »Urmel aus dem Eis«

4 Die Shell

Der Kommandointerpreter unter UNIX wurde von Anfang an Shell genannt. Verschiedene Shells wurden im Laufe der Jahrzehnte entwickelt, immer wieder verbessert und verändert. Unter Linux wurde die Shell stark erweitert und unter dem Namen bash (Bourne Again Shell) zur Standard-Shell.

Leistungsfähig

Die Leistungsfähigkeit der Shell führt dazu, dass erfahrene Linux-Anwender immer wieder die Maus zur Seite schieben und ein paar kurze Befehle per Tastatur eingeben. In vielen Fällen geht dies schneller als mit jeder grafischen Oberfläche.

Wenn Sie einen Server betreiben, werden Sie feststellen, dass viele Arbeiten von der Konsole aus schneller gehen. Insbesondere können die Kommandos zu Skripten zusammengefasst werden, die bestimmte Abläufe steuern. Ein wichtiger Aspekt ist, dass die Shell über das Netzwerk aufgerufen werden kann und damit eine vollständige Administration aus der Ferne möglich ist.

Shell-Spaltereien

Login-Shell

Welche Shell nach dem Einloggen gestartet wird, steht in der Datei */etc/passwd*[1]. Eine Shell, die auf diese Weise gestartet wird, bezeichnet man als interaktive Login-Shell. Sie können eine Shell aber auch direkt aus einer anderen Shell oder durch das Starten einer Terminalemulation der grafischen Oberfläche starten. Eine solche Shell ist keine Login-Shell, aber dennoch eine interaktive Shell.

Nicht interaktive Shell dash

Es gibt auch nicht interaktive Shells. Diese dienen als reiner Kommandointerpreter zur Ausführung eines Shell-Skripts. Mit dem Ende des Skripts endet auch die Shell. Unter Debian Squeeze wird dash als Nachfahre der ash als nicht interaktive Shell eingesetzt, da sie sehr schnell und klein ist.

1 siehe Abschnitt 13.2 Seite 417

Shell und POSIX Der POSIX-Standard fordert nach POSIX.2, dass jedes konforme Betriebssystem durch den Aufruf von `sh` eine POSIX-konforme Shell startet. Da POSIX ein Mindeststandard ist, wird diese Forderung von der komfortablen bash erfüllt.

4.1 Shell-Start

Im EDV-Raum stand früher am UNIX-Server immer noch ein Terminal, das über eine serielle Schnittstelle angeschlossen war. Diese ermöglichte auch dann noch den Zugang zum Server, wenn das Netzwerk zusammengebrochen war.

Sechs Konsolen Bei Linux sind solche Terminals bereits im System integriert. Wenn Sie vor der grafischen Oberfläche eines Linux-Systems sitzen, können Sie mit der Tastenkombination `Strg`+`Alt`+`F1` auf die erste Konsole umschalten. Wenn Sie `Alt`+`F2` drücken, kommen Sie auf eine weitere Konsole. Standardmäßig sind sechs Konsolen eingebaut. Durch die Tastenkombination `Strg`+`Alt`+`F7` kommen Sie wieder zurück in die grafische Umgebung.

Mit der Tastenkombination `Strg`+`Alt`+`F8` erhalten Sie eine Fehlerkonsole, auf der Sie die letzten Systemmeldungen finden, die Debian für berichtenswert hält.

Serielles Terminal Auf diese Methode können Sie eine Notadministration auch dann noch durchführen, wenn im Netzwerk gar nichts mehr geht. Theoretisch kann es Ihnen natürlich passieren, dass die grafische Oberfläche nicht mehr mitarbeiten will oder die Tastatur blockiert. Auch dagegen können Sie sich noch absichern. Falls Sie an Ihrem Server noch eine serielle Schnittstelle haben und auf dem Dachboden noch ein serielles Terminal finden, können Sie es auch heute noch an Ihren Debian-Server anschließen. Sollte Ihnen diese Hardware fehlen, können Sie mit einem gängigen USB-RS232-Adapter und einer Terminalemulation auf einem Notebook den gleichen Effekt erzielen. Ob Sie eine solche Umgebung aufbauen wollen, hängt ganz von Ihren persönlichen Sicherheitsbedürfnissen ab.

Grafisches Terminal Sehr viel wahrscheinlicher wird es sein, dass Sie aus der grafischen Oberfläche eine Terminalsitzung öffnen. Bei GNOME erreichen Sie über ANWENDUNGEN • ZUBEHÖR • TERMINAL eine Terminalemulation im Fenster. Der Vorteil dieser grafischen Terminals ist die Möglichkeit, beliebig viele davon aufrufen zu können. So haben Sie Ihre Terminalsitzungen direkt

nebeneinander und können sie sogar noch durch Farbgebung nach Tätigkeitsgebiet markieren.

Sie können sich auf Ihrem Server auch über das Netzwerk anmelden. Dazu verwenden Sie ssh[2]. Auf diesem Weg können Sie Ihren Server administrieren, obwohl Sie sich sehr weit davon entfernt befinden. Auch in diesem Fall arbeiten Sie über eine Kommandoeingabe der Shell.

Fernbedienung per ssh

4.2 Befehlsempfänger

Wenn Sie eine Terminalsitzung eröffnet haben, finden Sie an jedem Zeilenanfang einen sogenannten »Prompt«. Unter Debian steht darin standardmäßig der Hostname und der aktuelle Pfad. Es folgt ein Dollarzeichen bei normalen Benutzern oder ein Doppelkreuz, falls Sie als root angemeldet sind. Daneben blinkt der Cursor. Dort geben Sie Ihre Kommandos ein und schließen sie mit der Return-Taste ab. Diese Eingaben gehen an die Shell. Sie übernimmt die Interpretation der Befehle und ruft zur Ausführung das Betriebssystem oder die angesprochenen Programme auf.

4.2.1 Befehl, Optionen, Argumente

Ein Befehl beginnt mit dem Befehlsnamen. Dieser bezeichnet meist ein Programm. Es gibt allerdings auch Befehle, die die Shell selbst interpretiert.

Es folgen ein oder mehrere Leerzeichen, um die Parameter vom Befehlsnamen zu trennen. Auch die Parameter werden voneinander durch Leerzeichen getrennt. Parameter unterteilen sich in Optionen und Argumente.

Leerzeichen ist Trennzeichen

Optionen sind an einem Minuszeichen zu erkennen. Sie bewirken eine Veränderung der Programmausführung. Werden mehrere Optionen mitgegeben, können diese direkt hintereinandergeschrieben werden, und nur ein Minuszeichen muss am Anfang vergeben werden. Statt `-l -a` kann es also auch `-la` heißen.

Optionen

Neuere Programme verwenden gern Wörter als Optionen. Zur Unterscheidung von kombinierten Optionen benutzen sie meistens zwei Bindestriche[3].

Doppel-Minus

2 siehe Abschnitt 9.3 Seite 310
3 Eine Ausnahme bildet beispielsweise `find`, das Optionswörter, aber nur einen Bindestrich verwendet.

Argumente

Schließlich haben die meisten Befehle Argumente. Dies sind die Objekte, auf denen die Befehle ausgeführt werden sollen, meist Dateien oder Verzeichnisse. Je nach Art des Befehls kann es gar keine oder beliebig viele Argumente geben. Die Argumente werden durch Leerzeichen getrennt. Sie können auch durch sogenannte Wildcards beschrieben werden.[4]

[zB]

Um Befehl, Option und Argument etwas anschaulicher darzustellen, greifen wir auf den alten Schlager »Und dann hau ich mit dem Hämmerchen mein Sparschwein« von Chris Howland zurück. Auf einer Linux-Konsole würde Chris Howland es vielleicht so formulieren:

```
debian $ hau --haemmerchen sparschwein
```

Der Befehl `hau` wird aufgerufen. Die Option `--haemmerchen` besagt, auf welche Weise der Befehl ausgeführt wird und *sparschwein* ist das Argument, auf das der Befehl wirkt.

4.2.2 Befehlspfade

PATH

Externe Befehle sind Programme, die sich irgendwo im Verzeichnisbaum befinden können. Damit der Computer nicht bei jedem Befehl eine komplette Hausdurchsuchung der Festplatte ausführt, werden die Befehle nur in bestimmten Verzeichnissen abgelegt. Die Namen der Verzeichnisse werden in der Reihenfolge der Wichtigkeit in der Umgebungsvariablen PATH notiert. Die Verzeichnisnamen werden durch Doppelpunkte voneinander getrennt.[5]

Da die Pfade in einer Variablen stehen, können für Anwender und Administrator unterschiedliche Pfadfolgen verwendet werden. Jeder Benutzer kann seinen Pfad individuell ändern, indem er den Inhalt der Variablen im Startskript ändert.

Der Befehl which

which, whereis

Wenn Sie nachvollziehen wollen, welche Programme die Shell verwendet, können Sie die Befehle `whereis` und `which` zurate ziehen. Der Unterschied zwischen den Strategien ist meist nur geringfügig.

Der Befehl which: Welches Programm wird gestartet?
`which` *<Befehl>*

4 Wildcards siehe Abschnitt 4.2.3 Seite 108
5 siehe Abschnitt 4.7.2 Seite 132

Pfadbestimmung

Der Befehl `which` ermittelt, wo sich das Programm befindet, das als Argument angegeben wird. Wenn Sie einen Befehl eingeben, durchsucht die Shell die Befehlspfade anhand der Umgebungsvariablen PATH nach dem passenden Programm.

Das erste Programm im Pfad, das dem Namen entspricht, wird inklusive seines Pfades angezeigt. Wenn es also mehrere Programme gleichen Namens im Suchpfad gibt, zeigt das Programm `which` an, welches Programm davon ausgeführt wird. Im folgenden Beispiel wird der Pfad des Skripts `cddasi` angezeigt.

```
debian $ which cddasi
/home/arnold/bin/cddasi
```

Anhand des Pfades lässt sich also erkennen, dass es sich bei `cddasi` nicht um ein Systemprogramm handelt, sondern um ein Skript, das der Anwender selbst geschrieben hat.

Der Befehl whereis: Suchen nach einem Programm

`whereis` *<Befehl>*

Standardorte

Ein naher Verwandter dieses Befehls ist `whereis`. Er wird genauso aufgerufen wie `which` und liefert auch den Pfad eines Programms. Darüber hinaus zeigt er aber auch den Ort der Manpage. Das Programm `whereis` folgt nicht der Variablen PATH, sondern durchsucht die typischen Pfade, in denen das gesuchte Programm stehen könnte. Aus diesem Grund würde `whereis` das Skript `cddasi` nicht finden. Dafür findet der Befehl den Pfad von Administrationsprogrammen, die gar nicht im Pfad des normalen Benutzers stehen. Im folgenden Beispiel zeigt `whereis` sowohl den Pfad des Befehls `ifconfig`[6] als auch dessen Manpage, obwohl dieser Befehl nur im Pfad des Superusers root liegt und bei Aufruf durch normale Benutzer nicht gefunden würde.

```
debian $ whereis ifconfig
ifconfig: /sbin/ifconfig /usr/share/man/man8/ifconfig.8.gz
```

Der Befehl `which ifconfig` meldet tatsächlich keinen Pfad. Und auch der Aufruf von `ifconfig` erzeugt eine Fehlermeldung, da der Pfad */sbin* nicht in der Variablen PATH der normalen Anwender aufgeführt ist. Dagegen führt der Aufruf mit dem Pfad auch für den normalen Benutzer zu einem Ergebnis. Tatsächlich ist der Befehl für den Normalbenutzer sogar recht

6 Der Befehl `ifconfig` zeigt und ändert die Netzwerkschnittstellen des Systems (siehe Abschnitt 7.3.4 Seite 251).

hilfreich, wenn es darum geht, festzustellen, ob der eigene Computer korrekt ins Netzwerk eingebunden ist.

```
debian $ /sbin/ifconfig
eth0  Protokoll:Ethernet  Hardware Adresse 00:19:7D:4C:FD:0B
      inet Adresse:192.168.109.101  Bcast:192.168.109.255
...
```

4.2.3 Zugriff auf mehrere Objekte

Argumentliste

Sollen mehrere Dateien als Argumente für einen Befehl verwendet werden, lassen sie sich meist einfach aufzählen. Die meisten Kommandozeilenprogramme sind so geschrieben, dass sie beliebig viele Dateien als Argumente akzeptieren. Um die Aufzählung zu vereinfachen, können Sie durch eine Maske mehrere Dateien zusammenfassen. Die Shell sucht die auf die Maske passenden Dateien zusammen und übergibt sie dem aufgerufenen Programm als Liste.

Wildcards: *, ? und die eckigen Klammern

Stern

Um Dateien mit ähnlichen Namen zu ermitteln, wird meist der Stern als Platzhalter für die Zeichen verwendet, die sich unterscheiden. Der Stern steht als Ersatz für beliebig viele Zeichen. An der Stelle des Sterns kann auch gar kein Zeichen stehen. Die folgende Liste zeigt einige Beispiele für Argumente des Befehls `ls`:

- `ls prog*` – alle Dateien, die mit prog anfangen.
- `ls *mein` – alle Dateien, die mit mein aufhören.
- `ls OS*.c` – alle Dateien, die mit OS anfangen und mit .c aufhören.
- `ls *dat*` – alle Dateien, die dat im Namen enthalten.

Fragezeichen

Soll eine genaue Anzahl von Zeichen freigehalten werden, benutzt man das Fragezeichen. Es steht für genau ein Zeichen. `M??s` steht also für Maus, Mais oder Muks. Murks würde nicht passen, da die drei Buchstaben zwischen M und s nicht zu den zwei Fragezeichen passen. Auch eine Kombination aus Fragezeichen und Sternen kann sinnvoll sein. Wenn man beispielsweise alle Dateien und Verzeichnisse, die mit einem Punkt beginnen, löschen möchte,[7] sollte man lieber nicht `rm -r .*` eingeben. Da der Stern auch die leere Zeichenkette symbolisiert, würden damit das

7 Es ist übrigens keine gute Idee, diese Dateien aus dem Benutzerverzeichnis zu löschen. Sollten Sie es dennoch gerade getan haben, vergessen Sie bitte, wo Sie das gelesen haben.

aktuelle und das Elternverzeichnis ausgeräumt. Das Elternverzeichnis ist das davor liegende Verzeichnis und wird mit .. angesprochen. Besser ist da der Gedanke, mit `rm -r .??*` zu arbeiten. Damit werden dann weder . noch .. getroffen, da beide nicht aus drei Zeichen bestehen. Allerdings würde auch die Datei *.ab* erhalten bleiben.

Eckige Klammern

Neben diesen Wildcards gibt es noch die Möglichkeit, mit den rechteckigen Klammern gewisse Alternativen für ein Zeichen zu verwenden. Beispielsweise bedeutet `[Mm]akefile`, dass die Datei *makefile* oder *Makefile* heißen kann. Die Maske `[A-Z][0-9]?*` beschreibt alle Dateien, deren Namen mit einem Großbuchstaben beginnen. Darauf muss eine Ziffer folgen. Außerdem muss der Dateiname mindestens aus drei Zeichen bestehen.

Sonderzeichen als Parameter

Es kommt vor, dass ein Befehlsparameter ein Zeichen enthält, das von der Shell interpretiert wird. Um diesen an ein Programm zu übergeben, muss das Sonderzeichen vor der Interpretation durch die Shell geschützt werden.

Backslash

Bei einzelnen Zeichen verwenden Sie dazu einfach einen Backslash (\). Der deutsche Begriff dafür heißt wörtlich übersetzt »rückwärtiger Schrägstrich«. * steht für einen Stern, \? für ein Fragezeichen im Namen. Wenn ein Dateiname ein Leerzeichen enthält, muss auch diesem ein Backslash vorangestellt werden, damit die Shell nicht aus dem einen Namen zwei macht.

Anführungszeichen

Alternativ kann das Argument auch in Anführungszeichen (") oder Hochkommata (') gesetzt werden. In diesem Fall interpretiert die Shell nicht die Sonderzeichen, sondern reicht sie direkt an das aufgerufene Programm weiter. Der Unterschied zwischen beiden ist, dass Variablen in Anführungszeichen noch aufgelöst werden, in Hochkommata nicht.[8]

4.2.4 Fehler

Fehlernummern

Jedes Programm liefert nach seiner Fertigstellung Fehlernummern zurück. Systemweit einheitlich ist, dass die Rückgabe von Null dafür steht, dass das Programm fehlerfrei beendet wurde. Jede andere Nummer gilt als Fehler. Die Bedeutung der jeweiligen Fehlernummer legt allerdings jedes Programm selbst fest.

8 siehe Abschnitt 4.8.2 Seite 141

Fehlermeldung

Auf dem Bildschirm erscheinen im Fehlerfall Meldungen, die vom aufgerufenen Programm stammen. Die Shell meldet sich, wenn sie das aufgerufene Programm nicht kennt oder die Struktur des Befehls ihr nicht behagt. So könnte nach dem Aufruf von `abcdefg` eine Meldung wie die folgende erscheinen:

```
abcdefg: command not found.
```

Da es kein Programm namens `abcdefg` gibt, wird die Shell melden, dass sie diesen Befehl nicht ausführen konnte.

Fehlermeldung des Programms grep

Kommt ein korrekt aufgerufenes Programm mit der Eingabe nicht zurecht, meldet es sich. Zum Beispiel erscheint nach der Eingabe `grep o p` folgende Meldung:

```
debian $ grep suchwas datei
grep: datei: No such file or directory
debian $
```

Der Befehl `grep` sucht die Zeichenfolge »suchwas« in der Datei namens *datei*, die es offenbar nicht gibt.[9]

4.3 Kommandos verknüpfen

Kurz und knapp

Idealerweise erledigt jedes Programm genau seine Aufgabe und liefert eine Ausgabe, die von anderen Programmen wieder als Eingabe verwendet werden kann. Die Shell stellt über Pipes die Verbindung her. So ist es möglich, die Programme wie Legosteine miteinander zu verbinden.

4.3.1 Ein- und Ausgabe als Datenstrom

stdin und stdout

Die Tastatureingabe und die Bildschirmausgabe werden jeweils als Datei aufgefasst. Die Standardeingabe heißt *stdin* und die Ausgabe *stdout*. Die meisten Programme erwarten ihre Eingaben von *stdin* und schreiben ihre Ergebnisse nach *stdout*.

stderr

Es gibt einen zweiten Ausgabekanal, der für die Ausgabe von Fehlermeldungen verwendet wird und darum *stderr* heißt. Im Normalfall wird er wie *stdout* auf dem Bildschirm ausgegeben. Wenn Sie aber die Ergebnisse von Programmen zur Weiterverarbeitung umleiten, können Sie die Fehlermeldungen auf dem Bildschirm belassen. Dort stören sie nicht die

9 grep siehe Abschnitt 5.6.3 Seite 202

Weiterverarbeitung, und Sie können sofort erkennen, wenn ein Fehler auftritt.

Dateiende-kennung

Wenn die Tastatureingaben als Datei interpretiert werden, ist es von Zeit zu Zeit erforderlich, mit der Tastatur ein Ende der Datei zu simulieren. Dafür dient die Tastenkombination `Strg`+`D` am Anfang der Zeile. Wenn Sie dies in der Eingabeaufforderung einer Shell tun, geht diese vom Ende der Verarbeitung aus und beendet sich selbst.

4.3.2 Datenströme umleiten

Größerzeichen

Wenn Sie die Ausgabe eines Befehls in eine Datei umlenken wollen, hängen Sie das Größerzeichen an das Ende der Befehlszeile, gefolgt von dem Namen der Ausgabedatei.

```
debian $ ls -l > ausgabedatei
debian $
```

Zieldatei

Der Befehl `ls -l` zeigt die Langform aller Dateinamen im aktuellen Verzeichnis an. In dem Befehl wurde diese Ausgabe in die Datei namens *ausgabedatei* umgeleitet. Darum erscheint sofort der Prompt wieder. In der Datei finden Sie nun die Ausgabe des Befehls. Sie könnten sie weiter bearbeiten oder als Protokoll archivieren.

Eingabeumleitung

So wie es möglich ist, die Ausgabe umzuleiten, können Sie auch die Eingabe umleiten. Dazu verwenden Sie das Kleinerzeichen. Auch dies wird an das Ende des Befehls angehängt, gefolgt von der umzuleitenden Datei. Auf diese Weise können Sie bei Programmen, die mehrere Eingaben nacheinander einfordern, Ablaufszenarien vorbereiten und diese mit einem einzigen Befehl ablaufen lassen.

```
debian $ navi < TanteAnna
```

Im Beispiel wird dem Navigationsprogramm `navi` der Inhalt der Datei *TanteAnna* zugeführt. In dieser Datei könnte beispielsweise folgender Inhalt stehen:

```
61250
Usingen
Friedrich-Stengel-Str. 1
j
```

Diese fünf Zeilen antworten auf die Eingaben, auf die das Programm `navi` wartet: »Postleitzahl«, »Ort«, »Straße und Hausnummer«, »sofort starten«. Wie von Zauberhand geführt, wird das Programm automatisch weiterlaufen.

Sie können sowohl die Eingabe als auch die Ausgabe umleiten, wie das folgende Beispiel zeigt:

```
sort <eingabedatei  >ausgabedatei
```

Auf diese Weise wird die Datei *eingabedatei* als Eingabe für den Sortierbefehl verwendet und die Ergebnisse werden in der Datei *ausgabedatei* abgelegt.

Datei stutzen

Beim Umleiten einer Ausgabe wird die Zieldatei zunächst geleert. Diesen Effekt kann man nutzen, wenn eine Protokolldatei zu groß wird. Da solche Dateien von anderen Prozessen beschrieben werden, kann man sie nicht einfach löschen. Auch wenn man die Datei unter diesem Namen wieder neu erzeugt, ist es nicht dieselbe Datei, die der Hintergrundprozess bearbeitet hatte. Mit dem Größerzeichen und dem Dateinamen wird die Datei sofort auf 0 Byte zurückgesetzt.

```
> /var/log/messages
```

Ausgabe anhängen: >>

Nicht immer soll der Inhalt der Datei gelöscht werden, in die man die Ausgabe umleitet. Verwendet man statt eines Größerzeichens zwei, so wird die Ausgabe an die existierende Datei angehängt.

stderr umleiten: 2>

Um *stderr* umzuleiten, wird eine 2 vor das Größerzeichen geschrieben. Dies ist beispielsweise wichtig, wenn die Fehlermeldungen eines Compilers in einer Datei aufgefangen werden sollen.[10]

```
g++ mistprogramm.c 2>fehlerliste
```

Um *stdout* und *stderr* in die gleiche Datei umzuleiten, wird dem Umleitungszeichen ein kaufmännisches Und Zeichen vorangestellt.

```
g++ mistprogramm.c &> fehlerliste
```

Befehl	Wirkung
`g++ haus.c > out`	Umleitung der Standardausgabe, Fehler am Terminal
`g++ haus.c 2> out`	Umleitung der Fehler, Standardausgabe am Terminal
`g++ haus.c &> out`	Umleitung der Standardausgabe und der Fehler

Tabelle 4.1 Umleitungen

/dev/null

Manchmal entstehen Ausgaben, die nur stören. Um sie loszuwerden, können Sie sie einfach in die für solche Zwecke vorgesehene Datei */dev/null* umleiten. Alle Daten, die auf diese Pseudodatei umgeleitet werden, verschwinden auf Nimmerwiedersehen.

10 Das funktioniert nicht in der C-Shell.

4.3.3 Durch die Röhre schicken

Die Umleitung der Dateieingaben und -ausgaben kann auch so gestaltet werden, dass ein Programm die Ausgaben eines anderen Programms als Eingabe erhält. Eine solche Verknüpfung wird als »Pipe« bezeichnet. Syntaktisch wird ein senkrechter Strich zwischen den Aufruf der Programme gesetzt. Ein typisches Beispiel ist die Anzeige eines längeren Verzeichnisses, das zur leichteren Betrachtung seitenweise ausgegeben werden soll. Der Befehl `ls -l` erzeugt die Ausgabe, die an das Programm `more` weitergeleitet wird. Die Aufgabe von `more` ist es, den Dateneingang seitenweise anzuzeigen und jeweils zum Seitenende auf den Tastendruck des Anwenders zu warten.

```
ls -l | more
```

Seitenweises Blättern

Durch den senkrechten Strich | wird eine Pipe aufgebaut, die die Ausgabe des links stehenden Kommandos in die Eingabe des rechts stehenden Kommandos umleitet. Programme, die im Datenstrom einer Pipe verwendet werden, nennt man Filter.

Fehlerkanal

Die Pipe lenkt nur die Standardausgabe um. Der Fehlerkanal wird auf dem Bildschirm ausgegeben. Sie können aber ähnlich wie bei der Umleitung den Fehlerkanal und die Standardausgabe gebündelt in die Pipe senden, indem Sie hinter den senkrechten Strich ein kaufmännisches Und (&) stellen.

```
verarbeite |& protokolliere
```

Im obigen Beispiel würde das Ergebnis des Skripts `verarbeite` durch das Skript `protokolliere` übernommen werden, und zwar sowohl die Ausgabe als auch der Fehlerkanal.

Bindestrich als Dateiname

Einige Programme geben ihre Ergebnisse nicht auf *stdout* aus, sondern benutzen immer eine Ausgabedatei. Auch benannte Dateien können in eine Pipe umgeleitet werden. Dazu wird statt des Dateinamens ein Bindestrich verwendet. Umgekehrt kann auch statt der Eingabedatei ein Bindestrich verwendet werden.

Verzeichniskopie per tar

Ein Beispiel für eine solche Pipe ist das Kopieren eines Verzeichnisbaums durch das Datensicherungsprogramm tar[11]. Zunächst wird tar aufgerufen, um den aktuellen Verzeichnispfad in eine Datei zu sichern. In diesem besonderen Fall wird dies eine Pipe sein, also erhält der Befehl statt des Dateinamens einen Bindestrich. Es folgt der senkrechte Strich als Zeichen einer Pipe.

11 siehe Abschnitt 15.5 Seite 464

Zielwechsel

Nach dem Aufbau der Pipe wird in das Zielverzeichnis gewechselt. Dort soll der Datenstrom wieder in Dateien umgewandelt werden. Um den Inhalt der Pipe auszulesen, wird wiederum der Bindestrich verwendet. Hier der komplette Befehl, der den Inhalt des Verzeichnisses */home/paul* in das Verzeichnis */home/johannes* sichert:

```
debian $ cd /home/paul
debian $ tar cf - . | ( cd /home/johannes ; tar xfp - )
```

Datenabzweigung: tee

Ausgabe vervielfältigen

Die Daten, die per Pipe von einem zum anderen Programm gelangen, sind nach außen nicht sichtbar. Manchmal werden die Daten, die durch die Pipe geleitet werden, später noch einmal für andere Zwecke benötigt. In diesem Fall können Sie das Kommando `tee`, gefolgt von einem Dateinamen, in die Pipe einfügen. Dann wird der aktuelle Datenstrom in diese Datei kopiert, während die nächste Anwendung in der Pipe trotzdem den Datenstrom unverändert weiterverarbeiten kann. Beispiel:

```
debian $ ls | tee out | wc
```

Abzweigung

In der Datei *out* befinden sich nach der Ausführung des Befehls die Dateinamen des aktuellen Verzeichnisses, während die Ausgabe des Befehls die Anzahl der Dateien darstellt. Das Programm `tee` bildet also das T-Stück im Datenstrom und zweigt Daten in die angegebene Datei ab.

4.3.4 Quoting: Befehle verschachteln

Argumentliste

Bei der Pipe werden die Ergebnisse des Programms als Input eines anderen Programms verwendet. Sollen die Ergebnisse als Argumentliste verwendet werden, benutzt man das Quoting.

[zB]

Nehmen wir an, Sie möchten alle C++-Sourcen editieren, in denen die Funktion `TuDochWasEgon` verwendet wird. Der Befehl `grep`[12] listet mit der Option `-l` alle Dateien, die den übergebenen Suchbegriff enthalten. Diese Liste der Dateien könnte dem Editoraufruf als Parameter übergeben werden. Damit der Editor diese Liste als Parameter bekommt, wird der `grep`-Befehl in rückwärtige Hochkommata (backquotes) gesetzt.[13] Auf amerikanischen Tastaturen finden Sie den Backquote rechts neben dem Apostroph. Auf einer deutschen Tastatur befindet er sich rechts neben dem ß in Kombination mit der Hochstelltaste.

---

12 siehe Abschnitt 5.6.3 Seite 202

13 Von manchen Anwendern wird dieses Zeichen auch »Rückwärtsdüdel« genannt. Leider kann ich für die korrekte Schreibweise nicht garantieren.

```
debian $ vi `grep -l TuDochWasEgon`
```

Dateilisten

Dieser Mechanismus ist auch sehr praktisch, um Dateilisten anzufertigen, die von Skripten oder Befehlen verarbeitet werden sollen. Dazu schreiben Sie die betroffenen Dateinamen in eine Datei und lassen sie vom Befehl `cat` ausgeben.

```
debian $ verarbeite `cat Dateiliste`
```

cat als Quelle

In diesem Fall wird der Inhalt der Datei *Dateiliste* mit `cat`[14] ausgegeben und so als Argumentliste dem Skript `verarbeite` übergeben. Man könnte die Datei *Dateiliste* als Liste der Dateien zu einem Programmierprojekt benutzen. Um diese zu übersetzen, könnte der folgende Befehl abgesetzt werden:

```
debian $ g++ -o projekt `cat Dateiliste`
```

Eine Sicherung der Quelltexte könnte die gleiche Datei verwenden:

```
debian $ cp `cat Dateiliste` /usr/projekt/backup
```

Alternativschreibweise

Leider sind die Backquotes optisch kaum von einem Apostroph zu unterscheiden, haben aber eine völlig andere Bedeutung. Als Alternativschreibweise bietet die bash an, den in Backquotes stehenden Ausdruck einzuklammern und ein Dollarzeichen voranzustellen. Damit sind die beiden unten stehenden Ausdrücke gleichbedeutend:

```
debian $ verarbeite `cat filelist`
debian $ verarbeite $(cat filelist)
```

4.3.5 Von der Shell Prozesse steuern

Elternschaft

Wenn Sie von der Shell ein Programm aufrufen, wird die Shell gestoppt und wartet, bis das Programm endet. Anschließend steht sie für neue Abenteuer zur Verfügung. Mit dem Aufruf des Programms wird ein neuer Prozess geboren. Die Shell ist der Elternprozess, und jedes von der Shell gestartete Programm ist ihr Kindprozess. Damit hat jeder Prozess einen eindeutigen Elternprozess. Der Elternprozess wartet immer auf den Kindprozess und wertet dessen Rückgabewert aus.

Ampersand

In bestimmten Fällen möchten Sie aber dieses Warten durchbrechen und Prozesse im Hintergrund arbeiten lassen, während Sie im Vordergrund weiterarbeiten. Um ein Programm in den Hintergrund zu starten, setzen Sie einfach ein kaufmännisches Und (&) hinter den Aufruf. Dieses Zeichen wird schon wegen des holprigen deutschen Namens gern mit dem

14 cat siehe Abschnitt 5.6.1 Seite 201

englischen Begriff »Ampersand« bezeichnet. Das folgende Beispiel zeigt, wie ein Kompilierlauf[15] in den Hintergrund gestellt wird:

```
debian $ g++ meinprogramm.cpp &
[1] 13147
debian $
...
debian $
[1]+  Exit 1                g++ meinprogramm.cpp
debian $
```

Die Ausgaben dieses Kompilierlaufes werden allerdings weiterhin auf dem Terminal angezeigt, mit dem Sie weiterarbeiten. Das kann lästig werden, wenn im Vordergrund andere Programme laufen. Aus diesem Grund empfiehlt es sich, die Ausgaben von Hintergrundprozessen in Dateien umzuleiten.

```
debian $ g++ meinprogramm.cpp >out 2>err &
[1] 13147
debian $
...
debian $
[1]+  Exit 1                g++ meinprogramm.cpp >out 2> err
debian $
```

Jobs

Jobnummer Neben der PID führt die bash für jeden in den Hintergrund gestellten Prozess eine Jobnummer. Diese Nummern sind für den Anwender etwas übersichtlicher, weil sie immer wieder von vorn gezählt werden. Mit dieser Nummer können die Prozesse von der Shell aus angesprochen werden.

Prozessbeobachter Sie können Ihre Hintergrundprozesse betrachten, indem Sie den Befehl `jobs` verwenden. Sie erhalten eine Übersicht über die Prozesse, die Sie in den Hintergrund gestellt haben, die aber noch nicht abgelaufen sind.

```
gaston> jobs
[1]-  Running                 xman &
[2]+  Running                 xedit &
gaston>
```

Sie können sehen, dass hier die Programme `xman` und `xedit` in den Hintergrund gestellt worden sind. Es ist übrigens kein Zufall, dass beide

15 Ein Compiler übersetzt den Quelltext des Programmierers in ein ausführbares Programm.

Programme unter X laufen, also die grafische Oberfläche verwenden. Normalerweise wird man alle grafischen Programme von der Shell aus in den Hintergrund stellen. Während die Shell auf die normalen Kommandos warten wird, um ihr Ergebnis weiterzuverarbeiten, laufen die grafischen Programme in direkter Interaktion mit dem Benutzer, ohne die Shell zu benutzen. Da ist es am besten, Sie stellen diese Programme gleich in den Hintergrund.

Signale, die Prozesse morden

Programmende durch Strg+C

Ein von der Shell im Vordergrund gestarteter Prozess kann meist mit der Tastenkombination Strg+C abgebrochen werden. Der Prozess erhält dadurch das Terminierungssignal SIGINT. Sofern das Programm das Signal nicht abfängt und verweigert, stirbt der Prozess.

Programmunterbrechung durch Strg+Z

Sie können einen Prozess auch kurzfristig anhalten. Dazu verwenden Sie die Tastenkombination Strg+Z und erzeugen damit das Signal SIGTSTP. Es erscheint die Meldung, dass der Prozess gestoppt worden ist, und die Kommandozeile wird für weitere Eingaben frei:

```
debian $ xedit debian.tex
^Z
[1]+  Angehalten              xedit debian.tex
```

In der rechteckigen Klammer steht die Jobnummer aus Sicht der Shell, in diesem Fall eine 1. Diese Nummer darf nicht mit der PID verwechselt werden. Auf die Jobnummer beziehen sich die Kommandos `fg`, `bg` und `kill`, wenn ein Prozentzeichen vorangestellt wird.

Gestoppt

Der Prozess wurde durch das Signal SIGTSTP angehalten. Bei einer grafischen Anwendung erkennen Sie dies daran, dass das Fenster nicht aktualisiert wird, wenn Sie ein anderes Fester darüberlegen und anschließend wieder wegnehmen.

Fortsetzung folgt

Sie können nun entscheiden, ob Sie den gestoppten Job im Vordergrund oder im Hintergrund fortsetzen wollen. Sie können ihn auch vollständig stoppen. Wollen Sie den Prozess wieder durch die Shell kontrollieren, holen Sie ihn mit dem Kommando `fg` erneut in den Vordergrund:

```
debian $ fg %1
```

Vorn oder hinten

Damit wird die Situation, die vor dem Unterbrechen herrschte, wiederhergestellt. Das kann sinnvoll sein, wenn man kurz das Terminal für andere Zwecke benötigt. Soll der Prozess dagegen ab sofort im Hintergrund laufen, verwenden Sie das Kommando `bg`. Dieses Vorgehen ist ganz typisch, wenn man einen Prozess versehentlich im Vordergrund ge-

startet hat. Mit der Tastenkombination Strg+Z wird der Prozess kurz unterbrochen und mit dem Kommando `bg` in den Hintergrund geschickt.

```
debian $ bg %1
```

Letztendlich kann man den zunächst gestoppten Job auch terminieren:

```
debian $ kill %1
```

Prozente sind wichtig!

Insbesondere bei `kill` ist es wichtig, das Prozentzeichen vor der Zahl nicht zu vergessen, da diese Zahl die Jobnummer und nicht die PID bezeichnet. Es wird bitter, wenn es Ihnen gelingen sollte, den Prozess mit der PID 1 abzuschießen.

4.3.6 Anweisungen gruppieren

Sie können mehrere Befehle in einer Zeile aufrufen, wenn Sie ein Semikolon dazwischensetzen. Das Semikolon hat die gleiche Wirkung, als hätten Sie zwischen den beiden Kommandos die Enter-Taste verwendet. Das folgende Beispiel ruft zuerst den Befehl `make` auf und nach dessen Beendigung den Befehl `date`.

```
debian $ make; date
```

Hintereinander

Der Befehl `make`[16] wird vor allem von Programmierern verwendet, um komplexere Abläufe zu starten. Diese können manchmal recht lange dauern, und so kann auf die oben beschriebene Weise nach Abschluss der Arbeiten mithilfe des Befehls `date` angezeigt werden, wann der Durchlauf beendet wurde. `date` zeigt neben dem Datum auch die aktuelle Uhrzeit an.

Umleiten

Wenn Sie die Ergebnisse beider Befehle in einem Ausgabedatei umleiten wollen, werden Sie feststellen, dass die Umleitung nur auf den zweiten Befehl wirkt.

```
debian $ make; date > ausgabedatei
```

Dieses Verhalten ist logisch, wenn Sie das Semikolon als Ersatz für das Beenden des Befehls sehen. Eine Abhilfe wäre, wenn die beiden Programme jeweils eine Umleitung erhielten.

```
debian $ make > ausgabedatei; date >> ausgabedatei
```

Wichtig ist hier, dass der Befehl `date` zwei Größerzeichen verwendet, sonst würde die Ausgabe von `make` beim Start von `date` wieder gelöscht.

16 siehe Abschnitt 1.3.2 Seite 59

Klammersatz

Damit die beiden Kommandos zu einem zusammengefasst werden, können Sie eine Klammer um beide setzen. Statt der normalen Klammer können Sie auch eine geschweifte Klammer verwenden. Dann muss aber jeder Befehl mit einem Semikolon abgeschlossen werden. Im folgenden Beispiel sehen Sie den Unterschied in der Syntax.

```
( kommando1 ; kommando2 ) >ausgabedatei
{ kommando1 ; kommando2 ; } >ausgabedatei
```

Die Zusammenfassung gilt nicht nur für die Ausgabeumleitung, sondern auch für das Starten der Prozesse in den Hintergrund.

Feiner Unterschied

Es gibt einen feinen Unterschied zwischen den beiden Klammerarten. Die runden Klammern laufen in einer Subshell ab. Das heißt, dass beispielsweise Änderungen an den Umgebungsvariablen nach Verlassen der Klammer keine Wirkung mehr haben. Wie das folgende Skript zeigt, heißt Subshell nicht, dass ein neuer Interpreterprozess gestartet wird.[17]

```
HI="hi"
(echo $HI;KL=kl;echo $$)
{ echo $HI;GK=gk;echo $$; }
echo $KL $GK
```

Wird das Skript gestartet, geschieht Folgendes:

```
debian $ . klammer
hi
3979
hi
3979
gk
debian $ vi klammer
```

Sie sehen, dass der Inhalt von KL leer ist, obwohl die Variable in der runden Klammer belegt wurde. Sie sehen auch, dass auf die Variable HI sowohl innerhalb der geschweiften Klammer als auch innerhalb der runden Klammer zugegriffen werden kann, ohne dass man sie exportieren muss. Und Sie sehen, dass die PID in den beiden Klammerarten die gleiche ist.

Abhängigkeiten

Zwei Kommandos können nicht nur einfach getrennt werden. Es ist auch möglich, ein Kommando in Abhängigkeit davon auszuführen, ob ein anderes Kommando Erfolg hatte. So können Sie beispielsweise ein Dokument mit LaTeX generieren und dieses nur dann in PostScript konvertieren, wenn der erste Lauf erfolgreich war:

17 Umgebungsvariablen siehe Abschnitt 4.7.1 Seite 130

```
debian $ latex dokument.tex && dvips dokument
```

Andererseits gibt es auch den Fall, dass ein Kommando nur dann aufgerufen werden soll, wenn der vorherige Befehl erfolglos war. So können Sie eine Fehlermeldung ausgeben, wenn der Übersetzungslauf unbefriedigend war:

```
debian $ latex dokument.tex || echo "Schlimmer Fehler"
```

Syntax	Ausführung
prog1 ; prog2	Erst prog1, dann prog2 ausführen
prog1 && prog2	prog2 nur bei Erfolg von prog1 ausführen
prog1 || prog2	prog2 nur bei Misserfolg von prog1 ausführen

Tabelle 4.2 Start mehrerer Programme

4.4 History

Geschichtsschreibung

Sie können zuvor abgegebene Befehle zur Wiederverwendung in die Kommandozeile zurückholen. Dazu verwenden Sie die Pfeiltasten. Mit dem Pfeil oben holen Sie die letzten Befehle zurück. In der Datei *.bash_history* im Benutzerverzeichnis jedes Benutzers speichert die bash die letzten 1.000 eingegebenen Befehle. Die Datei speichert sie im Klartext. Zur Wahrung der Intimsphäre ist diese Datei für niemanden lesbar, außer dem Besitzer.

Alte Kamellen aufwärmen

Die Datei *.bash_history* dient nicht etwa dem Schmökern in alten Zeiten und Befehlen, sondern speichert die Befehle, um sie auf Anwenderwunsch wieder hervorzuzaubern. Das geschieht, sobald die Pfeiltaste mit dem Pfeil nach oben gedrückt wird. Bei jedem Tastendruck geht es eine Zeile weiter zurück in der Geschichte der Befehlseingaben. Die Pfeiltaste nach unten geht wieder in Richtung Zukunft.

4.4.1 Funktionstasten

Standardbelegung

Sind die alten Befehle erst einmal hervorgekramt, lassen sie sich nach Herzenslust verändern. Natürlich funktionieren auch die Pfeiltasten nach links und rechts. Eingegebene Buchstaben werden an der Cursor-Position eingefügt. Mit der Taste `Entf` kann das Zeichen unter dem Cursor gelöscht werden, und mit der Backspace-Taste wird das Zeichen links vom Cursor entfernt.

Aber es gibt noch mehr Steuerungsmöglichkeiten. Sie werden feststellen, dass Sie viele der Tastenkombinationen vom Editor emacs[18] her kennen.

Bewegung

Die Tastenkombination Strg+A bewegt den Cursor an den Anfang der Zeile, genau wie es die Taste Pos1 auch bewirkt. Wenn der Anfang mit Strg+A erreicht wird, überrascht es nicht, dass die Tastenkombination Strg+E den Cursor ans Ende der Zeile bringt. Das Löschen des Zeichens unter dem Cursor funktioniert, wie bereits gesagt, mit der Taste Entf. Alternativ klappt es auch mit Strg+D. Den Rest der Zeile löschen Sie mit der Tastenkombination Strg+K.

Vervollständigung

Ein besonderes Feature ist die Befehlsvervollständigung durch die Tabulatortaste. Wenn Sie die ersten Buchstaben eines Befehls eingeben und dann die Tabulatortaste drücken, wird der Befehlsname so weit vervollständigt, wie er eindeutig bestimmbar ist. Die Vervollständigung stockt an der Stelle, an der es mehrere Möglichkeiten gibt. Mit einem weiteren Druck auf die Tabulatortaste lernen Sie auch die Kandidaten kennen, die infrage kommen.

Argumentationshilfe

Was mit dem Befehl funktioniert, geht auch bei den Dateinamen, die als Argument des Befehls verwendet werden. Sie geben die ersten Buchstaben ein, drücken die Tabulatortaste, und die bash sucht aus den im aktuellen Verzeichnis vorliegenden Dateien diejenige aus, die passt. In den neueren Versionen geht es sogar so weit, dass die bash prüft, welche der Dateien zu dem Befehl passt. So sucht die bash nach dem Befehl `cd` nur Verzeichnisse heraus. Haben Sie `kpdf` eingegeben, werden Ihnen nur die PDF-Dateien aus dem Verzeichnis angeboten.

Umgebungsvariablen

Auch bei Umgebungsvariablen funktioniert es. Geben Sie `echo` ein Leerzeichen, ein Dollarzeichen und ein paar Buchstaben ein, vervollständigt die bash in gewohnter Art das Argument um alle denkbaren Umgebungsvariablen.

Auf der Suche

Die inkrementelle Suche verwendet wie emacs[19] die Tastenkombination Strg+R. Bei jedem Tastendruck wird in der History nach einem passenden Befehl gesucht. Dabei schlägt die bash immer den nächsten passenden Befehl vor.

```
debian $
(reverse-i-search)`gr': ps -ef | grep kppp
```

Signifikante Fragmente

Sie sehen im Beispiel, wie der Anwender »gr« eingegeben hat. Sie müssen also nicht mit dem Anfang der Zeile beim Suchen anfangen. Im Gegenteil,

18 emacs siehe Abschnitt 5.4.2 Seite 187
19 siehe Abschnitt 5.4.2 Seite 190

Sie sind schneller am Ziel, wenn Sie eine markante Zeichenfolge aus der Mitte wählen. Falls die angegebene Zeile es nicht ist, können Sie vor dem Weitertippen auch noch einmal Strg+R eingeben. Dadurch wird der nächstältere Befehl, der »gr« enthält, herausgesucht.

Hütchenspiel

Mit der Tastenkombination Strg+T tauschen Sie das unter dem Cursor stehende Zeichen mit dem links daneben stehenden. Dabei wird der Cursor um ein Zeichen weiter nach rechts versetzt.

Meta-Möglichkeiten

Wenn Sie auch noch die *Meta-Taste* verwenden, die auf PC-Tastaturen mit Alt bezeichnet ist, erhalten Sie noch ein paar weitere Möglichkeiten. So wird Alt+D das aktuelle Wort ab der Cursor-Position löschen. Mit Alt+T tauschen Sie das aktuelle Wort mit dem linken Nachbarn.

Taste	Funktion
Strg+A	An den Anfang der Zeile
Strg+E	An das Ende der Zeile
Strg+D	Löscht das Zeichen unter dem Cursor
Strg+K	Löscht bis an das Ende der Zeile
Strg+L	Löscht den Bildschirm, aber nicht die Eingabezeile
Strg+R	Inkrementelles Suchen
Strg+T	Tauscht das Zeichen mit dem linken Nachbarn

Tabelle 4.3 Funktionstasten in bash

4.4.2 vi-Kommandos

set -o vi

Sollten Ihnen die Kommandos von vi lieber sein, können Sie mit dem Befehl `set -o vi` dessen Tastenkombinationen aktivieren.

Kommandos editieren wie in vi

Danach schalten Sie mit der Esc-Taste in den Editiermodus um. Sie können nun mit + und – die vergangenen Zeilen wieder heranziehen, bis Sie zu dem Befehl gelangen, den Sie ausführen möchten. Hier können Sie mit dem Schrägstrich in den bisherigen Kommandos suchen. Haben Sie den gewünschten Befehl gefunden, können Sie ihn noch modifizieren. Innerhalb der Zeile können Sie mit `i` einfügen und mit `d` löschen. Das Bewegen innerhalb der Zeile funktioniert mit der Leertaste und Backspace, aber auch wortweise mit `w` und `b`. Wer mit `vi` umgehen kann, wird sich dabei wohlfühlen.

Kommandos	Funktion
^	An den Anfang der Zeile
$	An das Ende der Zeile
x	Löscht das Zeichen unter dem Cursor
d $	Löscht bis an das Ende der Zeile
/	Inkrementelles Suchen

Tabelle 4.4 Funktionskommandos der bash im vi-Modus

Wenn Sie wieder die emacs-Kommandos aktivieren wollen, geben Sie einfach den folgenden Befehl ein:

```
set -o emacs
```

4.4.3 C-Shell-History

Auch Freunde der C-Shell finden in der bash ihre History-Funktionen mit dem Ausrufezeichen implementiert. Dazu gelten die Befehle aus Tabelle 4.5.

Zeichen	Wirkung
!!	Ruft die letzte Zeile noch einmal auf
!5	Ruft die fünftletzte Zeile noch einmal auf
!abc	Ruft die letzte Zeile auf, die mit abc beginnt
!?abc	Ruft die letzte Zeile auf, die abc enthält
!\$	Verwendet das Argument der letzten Zeile an dieser Stelle

Tabelle 4.5 C-Shell-History

Teile des letzten Kommandos können ersetzt werden. Betrachten Sie dazu folgendes Beispiel:

```
hpsrv 2: lx hel*
lx: Command not found.
hpsrv 3: ^lx^ls
ls hel*
hello.cpp
hpsrv 4:
```

Das fehlerhafte `lx` wird durch das korrekte `ls` ausgetauscht. Nach Return wird der korrigierte Befehl gestartet. Im folgenden Beispiel wird noch einmal der Befehl geholt, der mit `ls` beginnt. Im Anschluss wird das

Argument des letzten Kommandos im neuen Kommando wieder verwendet:

```
hpsrv 4: !ls
ls hel*
hello.cpp
hpsrv 5: echo !$
echo hel*
hello.cpp
hpsrv 6:
```

4.5 Shell-Startdateien

Global: /etc/profile

Für diverse Voreinstellungen ist es sehr praktisch, dass die Shell bash beim Start einige Skriptdateien ausführt. Nach dem Einloggen werden zunächst die Befehle abgearbeitet, die in der Datei */etc/profile* stehen. Da diese Datei vom Systemadministrator verwaltet wird, werden zunächst also die systemweiten Vorgaben von allen Benutzern ausgeführt. Dazu zählen in der Voreinstellung vor allem die Umgebungsvariablen und die Werte für `ulimit`[20].

Im Benutzerverzeichnis

Danach führt die bash, wenn sie als Login-Shell gestartet wurde, die folgenden Dateien aus:

- *.bash_profile*: Wenn sie existiert, wird sie ausgeführt.
- Wenn die Datei nicht existiert, wird die Datei *.bash_login* ausgeführt.
- Existiert auch diese Datei nicht, wird die Datei *.profile* abgearbeitet.

Beim Abmelden wird die Datei *.bash_logout* ausgeführt, sofern sie vorhanden ist.

Diese Abläufe erfolgen nur nach dem Einloggen. Wenn Sie eine normale Konsole in einer grafischen Oberfläche starten, wird beispielsweise die Datei *.profile* gar nicht ausgeführt. Damit also Änderungen darin gültig werden, muss man sich noch einmal aus- und einloggen. Alternativ können Sie innerhalb einer Shell den Befehl `su` mit einem Minuszeichen, gefolgt von einem Leerzeichen und dem Benutzernamen, aufrufen.

Bei jedem Shell-Start

Die Datei *.bashrc* des Benutzerverzeichnisses wird dagegen jedes Mal ausgeführt, wenn eine neue Konsole gestartet wird.

20 ulimit siehe Abschnitt 4.5.2 Seite 126

Pfadergänzung

Einige Anwender legen sich gern eigene Skripte oder selbst geschriebene Programme in ein eigenes Verzeichnis namens *bin* innerhalb ihres Benutzerverzeichnisses. Um diese Befehle ohne explizite Pfadangaben verwenden zu können, würde man die lokale Datei *.bashrc* durch den folgenden Befehl ergänzen:

```
export PATH=$PATH:~/bin
```

4.5.1 alias

Kurzkommandos

Sie können mit dem Befehl `alias` ein neues Kommando, ein sogenannter Alias, für komplexere Befehle erzeugen. Damit können Abkürzungen für oft verwendete längere Befehle angelegt werden. Sehr beliebt ist die Verwendung eines einfachen »l« oder »ll« für das Komando `ls -l`. Dazu schreiben Sie in die Datei *.bashrc*: Folgendes definiert:

```
alias ll='ls -l'
```

Parameter

Nun wird jedes Mal, wenn `ll` als Befehl an der Konsole eingetippt wird, diese Zeichenkette durch `ls -l` ersetzt. Es ist durchaus möglich, an `ll` auch Parameter anzuhängen. So kann die Ausgabe nach dem Entstehungszeitpunkt sortiert werden.

```
debian $ ll -rt *.cpp
```

Dieser Befehl wird vom Aliasmechanismus in folgenden Befehl aufgelöst:

```
ls -l -rt *.cpp
```

alias aufheben

Um einen Alias wieder aufzuheben, wird der Befehl `unalias` verwendet.

```
debian $ unalias ll
```

alias anzeigen

Um festzustellen, ob der Befehl, den Sie gerade verwenden, vielleicht ein Alias ist, verwenden Sie den Befehl `type`. Er zeigt auch an, wie der Alias definiert wurde.

```
debian $ type ll
ll is aliased to `ls -l'
```

Handelt es sich bei dem Befehl nicht um einen Alias, wird der Pfad der Programmdatei angezeigt. Mit dem Befehl `file`[21] können Sie dann weiter feststellen, ob es sich um ein Skript oder um ein kompiliertes Programm handelt.

21 siehe Abschnitt 5.3.6 Seite 174

4.5.2 ulimit

Grenzen aufzeigen

Mit dem Befehl `ulimit` können den Anwendern Beschränkungen auferlegt werden, die die Größen von einigen Ressourcen betreffen. Der Befehl wird typischerweise in der Datei */etc/profile* vom Administrator gesetzt. Die gesetzten Begrenzungen können Sie sich als Anwender mit dem Befehl `ulimit -a` ansehen.

Core-Dump

Durch den Befehl `ulimit` kann auch die Größe des Core-Dumps beschränkt werden. Der Core-Dump stellt einen Speicherabzug dar, der erzeugt wird, wenn ein Programm abstürzt. Die heutigen Systeme sind so eingestellt, dass sie gar keinen Core-Dump mehr erzeugen. Ob das daran liegt, dass die Abstürze so selten sind, oder eher daran, dass inzwischen deutlich mehr Anwender als Programmierer Linux verwenden, bleibt ganz Ihrer Spekulation überlassen. Falls Sie aber Programmierer sind, wird Sie vor allem interessieren, wie die Begrenzung des Core-Dumps aufgehoben werden kann. Verwenden Sie dazu einfach folgenden Befehl:

```
ulimit -c unlimited
```

Die Optionen von `ulimit` sind in Tabelle 4.6 aufgeführt.

Option	Wirkung
`-a`	Anzeige aller aktuellen Limits
`-c`	Maximale Größe eines Core-Dumps
`-d`	Maximale Größe des Datensegments eines Prozesses
`-f`	Maximale Größe einer Datei
`-n`	Maximale Anzahl offener Dateien (nicht bei allen Systemen)
`-s`	Maximale Größe des Stacks eines Prozesses
`-t`	Maximale CPU-Zeit in Sekunden
`-v`	Maximale Größe des virtuellen Speichers

Tabelle 4.6 Optionen von ulimit

4.6 Shell-Skripte

Die Shell als Interpreter

Die Shell ist nicht nur einfach ein Befehlsempfänger, sondern auch eine durchaus leistungsfähige Programmiersprache. Zusammen mit der Kombinierbarkeit der Kommandos entsteht die Möglichkeit, auch komplexere Abläufe zu programmieren. Insbesondere im Bereich der Systemverwaltung ist diese Sprache einer Programmiersprache wie C sogar überlegen,

weil sie direkten Zugriff auf alle Kommandos hat und ohne großen Aufwand Programme starten und deren Ergebnisse auswerten kann.

Einsatz

Shell-Skripte ermöglichen es, komplexere Abläufe durch einen einfachen Aufruf zu erledigen. Dazu gehören Administrationsarbeiten, die zeitversetzt durch `cron`[22] oder `at`[23] gestartet werden. Auch Aufgaben, die per `sudo`[24] an Anwender weitergegeben werden sollen, können in Shell-Skripten festgelegt werden. Als rc-Skripte, die beim Booten gestartet werden, sind sie bei der Konfiguration des Systems erforderlich.

Erstellen und Starten eines Shell-Skripts

Editor

Um ein Shell-Skript zu erzeugen, starten Sie einen Editor und führen einfach ein paar Kommandos hintereinander zeilenweise auf. Beispielsweise steht in der Datei *skripttest*:

```
# Mein erstes Skript
echo "ach ja"
ls
echo "soso"
```

Ausführungsrechte setzen

Um die Textdatei wie ein Programm aufrufen zu können, reicht es, sie mit dem Befehl `chmod 755`[25] ausführbar zu machen. Anschließend können Sie das Skript durch Eingeben des Dateinamens direkt starten:

```
debian $ chmod 755 skripttest
debian $ ./skripttest
ach ja
skripttest
soso
debian $
```

Punkt Slash

Sie müssen vor den Aufruf von »skripttest« noch einen Punkt und einen Schrägstrich setzen. Das hängt damit zusammen, dass in der Umgebungsvariablen PATH der Punkt nicht aufgeführt ist. So können Programme im aktuellen Verzeichnis nur dann ausgeführt werden, wenn der Benutzer bewusst aus dem aktuellen Verzeichnis eben durch Voranstellen von ./ aufruft.

Subshells aufrufen

Direkter Aufruf

Angenommen, in der Datei *skripttest* steht ein Shell-Skript, das nur darauf

22 cron siehe Abschnitt 11.3 Seite 366
23 at siehe Abschnitt 11.4 Seite 367
24 sudo siehe Abschnitt 13.4 Seite 429
25 siehe Abschnitt 5.3.3 Seite 165

wartet, ausgeführt zu werden. Um es zu starten, können Sie verschiedene Wege gehen. Der erste Weg wurde schon beschrieben. Sie ändern die Dateirechte auf ausführbar und starten dann das Skript wie ein normales Programm:

```
debian $ ./skripttest
```

Shell-Aufruf mit Parametern

Der zweite Weg, ein Skript auszuführen, besteht darin, eine Shell aufzurufen und ihr die Datei als Parameter zu übergeben. Tatsächlich sind diese beiden Arten, ein Shell-Skript zu starten, äquivalent. Auch bei dem direkten Start des Dateinamens wird eine neue Shell (Subshell) gestartet, die die Datei *skripttest* interpretiert:

```
debian $ bash skripttest
```

Punkt

Mit dem Kommando . (Punkt) und dem Dateinamen kann man die Datei *skripttest* von der aktuellen Shell ausführen lassen:

```
debian $ . skripttest
```

Beim Punkt interpretiert die aktuelle Shell

Der Unterschied zwischen dem Aufruf per Punkt und dem Aufruf per Subshell ist wichtig, da ein mit dem Punkt aufgerufenes Skript keine neue Shell startet. Stattdessen interpretiert die laufende Shell das Skript. Nur so kann ein Skript den Zustand der aktuellen Shell verändern. Wechselt das Skript das Verzeichnis, legt es Umgebungsvariablen an oder verändert es Variablen, bleiben diese Änderungen auch nach dem Ende des Skripts gültig, da das Skript die aktuelle Arbeitsshell verwendet. Wird dagegen eine Tochter-Shell gestartet, wirken sich die Änderungen nur dort aus und haben keinerlei Auswirkungen auf die aktuelle Shell. Anders ausgedrückt: Soll ein Skript Variablen setzen, die in der aktuellen Shell später gebraucht werden, muss dieses Skript zwingend mit dem Punkt aufgerufen werden.

source

Da der Punkt nicht gut zu lesen ist, kann man stattdessen auch den Befehl `source` verwenden. Der Name veranschaulicht vielleicht noch mehr die Bedeutung des Befehls, nämlich dass der Inhalt der Datei als Quelltext in die aktuelle Shell mit einfließt. Vor allem, wenn in Shell-Skripten Variablendefinitionen in andere Skripte ausgelagert werden soll, sollte man den Befehl `source` statt des Punkts verwenden.

```
debian $ source skripttest
```

Ein Skript, der mit dem Punkt oder mit dem Befehl `source` aufgerufen wird, braucht auch kein Ausführungsrecht zu haben, da die Datei ja nicht ausgeführt wird, sondern nur ihr Inhalt als Quelltext gelesen wird.

Kommentieren

Das Kommentarzeichen in einer Skriptdatei ist #. Alles, was in der gleichen Zeile dahinter steht, geht den Interpreter nichts an.

Den Interpreter festlegen

Erste Zeile im Kommentar

In der ersten Zeile eines Skripts kann festgelegt werden, welche Shell bzw. welcher Interpreter für diese Datei geladen werden soll. Das erste Zeichen ist ein Kommentarzeichen. Es folgt ein Ausrufezeichen. Dann wird der Interpreter mit komplettem Pfad genannt. Beispiel:

```
#!/bin/sh
```

Die Shell sh ist der Name der POSIX-kompatiblen Minimal-Shell. Ursprünglich bezeichnete sh eine Bourne-Shell. In der aktuellen Debian-Version Squeeze ist sh ein Link auf dash. dash ist eine abgespeckte Shell, die auf die Verwendung als Interpreter optimiert ist.

Sollten Sie im Skript Besonderheiten der bash-Shell verwenden, würden Sie an dieser Stelle auf der bash-Shell als Interpreter bestehen:

```
#!/bin/bash
```

Statt einer Shell können Sie auch beliebige andere Interpreter aufrufen. Typische Beispiele sind Perl oder Python. Im Falle von Perl würde der Eintrag folgendermaßen lauten:

```
#!/usr/bin/perl
```

Zeilen umbrechen

Backslash: \

Wenn ein Shell-Kommando länger als eine Zeile wird, können Sie den Befehl aufspalten, indem Sie einen einzelnen Backslash (\) an das Ende der Zeile setzen. Achten Sie darauf, dass nicht ein Leerzeichen hinter dem Backslash steht. Im folgenden Beispiel werden vier Dateien durch den Befehl `cat`[26] auf dem Bildschirm ausgegeben:

```
cat datei1 datei2 \
    datei3 datei4
```

4.7 Variablen

Speicher und Name

Variablen werden verwendet, um Informationen unter einem Namen zu speichern und wieder abrufbar zu machen. Ein typisches Beispiel ist die

26 siehe Abschnitt 5.6.1 Seite 201

Variable PWD, die den aktuellen Verzeichnisnamen enthält, oder die Variable EDITOR, in der abgelegt wird, welchen Editor der Benutzer als Standard verwendet. Sie können auch selbst Variablen verwenden, um bestimmte Informationen festzuhalten. Dazu müssen Sie festlegen, welchen Namen Ihre Variable haben soll. Der Name von Variablen beginnt mit einem Buchstaben und kann dann beliebig viele Ziffern und Buchstaben enthalten. Auch der Unterstrich ist zulässig.

Füllen und Auslesen

Sie weisen einer Variablen einen Inhalt zu, indem Sie dem Variablennamen ein Gleichheitszeichen folgen lassen und dahinter den Wert angeben, den die Variable in Zukunft haben soll. Dabei darf kein Leerzeichen zwischen der Variablen, dem Gleichheitszeichen und dem zugewiesenen Wert stehen. Im folgenden Beispiel erhält die Variable INFO den Inhalt »Tolles Wetter!«.

```
INFO="Tolles Wetter!"
```

Auslesen

Sollte der Wert wie in diesem Fall Leerzeichen enthalten, schließen Sie ihn wie oben zu sehen in Anführungszeichen ein. Um den Inhalt einer Variablen auszulesen, stellen Sie dem Variablennamen ein Dollarzeichen voran. Zur Ausgabe von Variablen auf der Standardausgabe wird der Befehl `echo` verwendet:

```
echo $INFO
```

Variablendefinition

Eine Variable wird definiert, indem ihr ein Inhalt zugewiesen wird. Entsprechend wird sie wieder entfernt, wenn man ihr einen leeren Inhalt zuweist.

Variablen in Großbuchstaben

Übrigens müssen Variablennamen nicht zwingend in Großbuchstaben gesetzt sein. Da die vom System vorgegebenen Variablennamen groß sind, halten sich auch die meisten Anwender an diese Vorgabe. Der Übersichtlichkeit wegen sollte man hier nicht ohne zwingenden Grund eine andere Konvention einführen.

4.7.1 Shell- und Umgebungsvariablen

Shell-Variablen

Wenn Sie Variablen in Skripten oder auch in der normalen Shell verwenden, gelten diese Variablen nur innerhalb des Skriptes oder in der aktuellen Shell. Man spricht darum auch von Shell-Variablen. Rufen Sie ein anderes Programm oder ein weiteres Skript auf, sind die Shell-Variablen dort unbekannt. Wollen Sie erreichen, dass auch nachfolgende Programme oder Shell-Skripte den Inhalt der Variablen auslesen können, verwenden Sie den Befehl `export`.

```
debian $ MYENV="Tolles Wetter"
debian $ export MYENV
```

In der bash kann das Setzen und Exportieren der Variable in einem Kommandoschritt ausgeführt werden:

```
debian $ export MYENV="Tolles Wetter"
```

Damit wird die Shell-Variable MYENV zur Umgebungsvariablen (engl. *environment variables*), die von jedem Prozess gelesen werden kann, der von dieser Shell aus gestartet wird.

Variablenliste

Sie können sich eine Liste der Umgebungsvariablen mithilfe des Befehls `env` anzeigen lassen. Dabei werden Sie feststellen, dass Debian recht viele Umgebungsvariablen verwendet. Damit Sie überhaupt eine Chance haben, etwas zu finden, sollten Sie die Ausgabe durch `more`[27] schicken oder am besten gleich mithilfe des Befehls `grep`[28] nach den gesuchten Variablen filtern. Das folgende Beispiel zeigt, wie alle Umgebungsvariablen angezeigt werden, die die Silbe LANG enthalten:

```
debian $ env | grep LANG
INOLANG=de_de
LANG=de_DE@euro
debian $
```

4.7.2 Vordefinierte Umgebungsvariablen

LANG

Die Variable LANG wird auch als Language-Variable bezeichnet, weil mit ihr eingestellt wird, in welcher Sprache das System mit Ihnen kommuniziert. Durch Umschalten der Sprache gibt beispielsweise der Befehl `date` Datum und Uhrzeit in unterschiedlichen Sprachen und Formaten wieder.

```
debian $ date
Mo 17. Mai 21:49:25 CEST 2010
debian $ export LANG=da_DK
debian $ date
man maj 17 21:51:38 CEST 2010
debian $ export LANG=en_US
debian $ date
Mon May 17 21:52:07 CEST 2010
debian $
```

Wenn Sie in Ihrer Anmelde-Shell beispielsweise durch einen Eintrag in der Datei *.profile* die Umgebungsvariable LANG auf Dänisch umstellen,

27 more siehe Abschnitt 5.6.2 Seite 202
28 grep siehe Abschnitt 5.6.3 Seite 202

werden Sie das Gefühl haben, dass Sie einen dänischen Computer vor sich haben, obwohl alle anderen Benutzer dieses Computers weiterhin deutsche Meldungen erhalten. Das hängt damit zusammen, dass alle Prozesse Nachfahren der Anmelde-Shell sind und deren Umgebung erben.

PATH

Die Variable PATH enthält die Liste aller Verzeichnisse, auf denen die ausführbaren Programme zu finden sind. Die Verzeichnisse sind durch einen Doppelpunkt voneinander getrennt.

```
debian $ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Wenn Sie auf diesem Computer also ein Programm aufrufen, wird es die folgenden Verzeichnisse durchsuchen:

- /usr/local/sbin
- /usr/local/bin
- /usr/sbin
- /usr/bin
- /sbin
- /bin

PATH erweitern

Wenn Sie beispielsweise möchten, dass zudem die Programme im Verzeichnis */usr/games* durchsucht werden, können Sie den folgenden Befehl geben:

```
debian $ export PATH=$PATH:/usr/games
```

Punkt im PATH

Eine besondere Rolle spielt der Punkt in der Liste der Verzeichnisse. Nur wenn der Punkt in der Liste auftaucht, wird auch das aktuelle Verzeichnis nach Befehlen durchsucht. In früheren UNIX-Versionen war der Punkt immer bei normalen Anwendern gesetzt. Nur beim Administrator root hatte man ihn aus der PATH-Variablen entfernt. Inzwischen wird er auch bei normalen Benutzern nicht mehr aufgeführt. Das hat zur Konsequenz, dass Sie auch beim Aufruf eines Programms, das sich im aktuellen Verzeichnis befindet, dem Befehl den aktuellen Verzeichnispfad voranstellen müssen. Sie können dazu entweder den kompletten Pfad angeben oder verkürzt den einfachen Punkt als Zeichen für das aktuelle Verzeichnis. Da zwischen dem Pfad und dem Befehl immer ein Schrägstrich stehen muss, wäre das Voranstellen von `./` die kürzeste Form. Ein Programm im aktuellen Arbeitsverzeichnis mit dem Namen *prog* würden Sie also durch `./prog` aufrufen können.

Reihenfolge

Die Reihenfolge der Verzeichnisse in der Variablen PATH ist ebenfalls von Interesse, da in dieser Reihenfolge die Verzeichnisse daraufhin durchsucht werden, ob das aufgerufene Programm dort zu finden ist. Wird eines gefunden, wird es gestartet und die Suche abgebrochen.

PS1 bestimmt den Prompt

PS1 ist der Prompt, also die Zeichenkette, die im Normalfall links neben dem Cursor steht. Standardmäßig steht dort für den normalen Anwender ein Dollarzeichen ($) und für den Administrator ein Hashzeichen (#). Inzwischen ist es aber Mode geworden, dort alle möglichen Informationen abzulegen. Dadurch füllt auf manchem System der Prompt zwei Drittel der Zeile. Andererseits ist es sicher nicht verkehrt, wenn man darüber informiert wird, wer man eigentlich ist und wo man sich gerade befindet. Sie können sich aus dem Kürzel für den Anwender (`\u`), den lokalen Rechner (`\h`) und das aktuelle Verzeichnis (`\w`) einen eigenen Prompt basteln.

```
debian $ PS1="\u@\h:\w >"
arnold@debian:~/my/texte/tex/buch >
```

PS2

Die Variable PS2 bestimmt den Sekundärprompt. Er wird erzeugt, wenn das angefangene Kommando nach dem Return noch nicht beendet ist. Dies merkt die Shell bei Anführungszeichen oder Klammern, die gesetzt wurden, aber deren Gegenstück nicht eingegeben wurde. PS2 ist normalerweise ein einfaches Größerzeichen und wird nur selten verändert.

HOME

In der Variablen HOME steht der Pfad des Benutzerverzeichnisses des Anwenders, unter dessen User-ID das Shell-Skript läuft.

Sie können das Benutzerverzeichnis auch durch die Tilde ˜ abkürzen. Beginnt ein Verzeichnis mit einer Tilde ˜, wird sie durch den Pfad des Benutzerverzeichnisses ersetzt. *˜/test* wird die Shell zu */home/arnold/test* auflösen, wenn das Benutzerverzeichnis */home/arnold* ist.

PWD OLDPWD

In der Variablen PWD findet sich das aktuelle Arbeitsverzeichnis. Das zuletzt verlassene Verzeichnis finden Sie in der Variablen OLDPWD.

EDITOR

EDITOR enthält den Standardeditor, der von manchen Programmen gestartet wird, wenn der Benutzer Textdateien bearbeiten soll. Diese Variable kann der Benutzer an eigene Bedürfnisse anpassen, wenn er beispielsweise lieber vim statt nano benutzt.

TERM

In der Variablen TERM wird der Bezeichner der Terminalemulation gespeichert, unter der die Shell derzeit läuft. Unter Linux finden Sie normalerweise zwei Inhalte. Wenn Sie einen grafischen Desktop verwenden, lautet der Inhalt von TERM `xterm`. Wenn Sie dagegen ohne grafische Oberfläche arbeiten, enthält die Variable den Inhalt `linux`. Serielle Ter-

minals sind oft mit dem Uralt-Terminal VT-100 kompatibel. In Fällen, in denen Sie weder mit `linux` noch mit `xterm` gute Ergebnisse erzielen, könnte `vt100` vielleicht weiterhelfen.

LOGNAME USER

In den Variablen LOGNAME oder USER finden Sie den derzeit angemeldeten Benutzer.

Variable	Inhalt
EDITOR	Der Standardeditor
HOME	Benutzerverzeichnis des Benutzers
IFS	Alle Zeichen, die wie ein Leerzeichen wirken sollen
LANG	Codierung der Landessprache
LOGNAME	Der angemeldete Benutzer
PATH	Verzeichnisse, die nach Programmen durchsucht werden
PS1 PS2	Promptzeichen
PWD	Das aktuelle Arbeitsverzeichnis
TERM	Die Terminalemulation

Tabelle 4.7 Vorbelegte Umgebungsvariablen

4.7.3 Mit Variablen rechnen

Die folgende Zuweisung belegt die Variable `dieZahlPi` mit einem Wert:

```
dieZahlPi=3.14
```

Textvariable

Wenn Sie den Befehl `echo \$dieZahlPi` aufrufen, so entspricht die Ausgabe genau dem, was Sie den Variablen zugewiesen haben. Bevor aber zu hohe Erwartungen entstehen, ist zu sagen, dass `dieZahlPi` keineswegs eine Zahl ist. Sie ist nur die Zeichenkette »3.14«. Sie werden spätestens dann enttäuscht sein, wenn Sie versuchen, das Ergebnis einer Berechnung in einer Variablen abzulegen.

```
debian $ SUMME="2*2+4"
debian $ echo $SUMME
2*2+4
```

Die Variable enthält leider nur das, was Sie ihr übergeben haben. Die Anführungszeichen sind übrigens erforderlich, damit der Stern nicht als Dateimaske interpretiert wird.

bc

Um dennoch das Ergebnis zu berechnen, können Sie das Programm `bc` verwenden:

```
debian $ echo 2*2+4 | bc
8
```

Um das Ergebnis einer Variablen zuzuweisen, können Sie die Backquotes[29] verwenden:

Backquotes

```
debian $ WERT=`echo 2*2+4 | bc`
debian $ echo $WERT
8
```

Auch das Programm `expr` ermöglicht die Berechnung von Ausdrücken, ist allerdings auf ganze Zahlen beschränkt. Der Befehl erwartet als Parameter einen numerischen Ausdruck wie etwa 3 + 3. Dabei ist darauf zu achten, dass zwischen den Zahlen und den Operatoren ein Leerzeichen steht. Das Ergebnis solcher Rechenkünste können Sie ebenfalls über den Mechanismus der Backquotes einer Variablen zuweisen. Hier wird die alternative Schreibweise verwendet, in der der Befehl eingeklammert wird und ein Dollarzeichen vorangestellt wird:

expr

```
debian $ WERT=$(expr 2 \* 2 + 4)
```

Der Backslash vor dem * sorgt dafür, dass die Shell ihn nicht durch alle Dateinamen des Verzeichnisses ersetzt. Mit dem vorangestellten Backslash wird der Stern an den Befehl `expr` durchgereicht. In Tabelle 4.8 steht, welche Operatoren für `expr` verwendet werden können.

Die in Shell-Skripten verwendbaren Operatoren unterscheiden sich wenig von denen anderer Programmiersprachen.

Zeichen	Bedeutung
+	Addition
–	Subtraktion
*	Multiplikation
/	Division
%	Modulo: Rest einer Division

Tabelle 4.8 Operatoren in expr

Die bash hat von der Korn-Shell den Befehl `let` geerbt, der diese Berechnungen ein wenig vereinfacht: Sie stellen den Befehl `let` einer Variablenzuweisung voran. Als Operanden können +, –, *, / und % (als Modulo) verwendet werden. Auch Klammern werden korrekt interpretiert. Es darf allerdings kein Leerzeichen in dem Ausdruck stehen. Besonders praktisch

let

29 siehe Abschnitt 4.3.4 Seite 114

ist die Fähigkeit, mit verschiedenen Zahlensystemen zu arbeiten. Dazu wird einer Zahl die Basis, gefolgt von einem Hashzeichen (#), vorangestellt. Die Dualzahl 2#11 entspricht einer dezimalen 3, und um die Hexadezimalzahl 1a darzustellen, wird 16#1a oder 16#1A verwendet. Beispiel:

```
debian $ let WERT=45+5
debian $ echo $WERT
50
debian $ WERT=45+5
debian $ echo $WERT
45+5
debian $ let WERT=16%5
debian $ echo $WERT
1
debian $ let WERT=(1+3)*2
debian $ echo $WERT
8
debian $ let WERT=16#1a
debian $ echo $WERT
26
debian $
```

Alternative Doppelklammer

In der bash können arithmetische Ausdrücke in eine Doppelklammer gesetzt werden. Aber auch hier sind nur ganze Zahlen möglich.

```
debian $ WERT=$((45+5))
debian $ echo $WERT
50
debian $ WERT=$((16%5))
debian $ echo $WERT
1
debian $ WERT=$(((1+3)*2))
debian $ echo $WERT
8
debian $
```

4.7.4 Auf die Parameter zugreifen

Aufrufparameter

Sie können auch ein Shell-Skript mit Parametern aufrufen. Aus dem Shell-Skript heraus können Sie die Parameter über Pseudovariablen auslesen. Die Pseudovariable `\$0` enthält den Dateinamen, unter dem das Skript gestartet wurde. `\$1` enthält den ersten Parameter, `\$2` den zweiten und so weiter. Aus der Variablen `\$#` erfährt das Skript, mit wie vielen Parametern es aufgerufen wurde. Beispiel:

```
# Skript zeige
echo "Ich heiße " $0 " und habe " $# "Parameter"
echo $2 $1
```

Listing 4.1 Aufrufparameter lesen

Das Skript soll `zeige` heißen. Nachdem Sie mit `chmod` die Rechte auf ausführbar gesetzt haben, können Sie das Programm aufrufen. Das Ergebnis sieht so aus:

```
debian $ zeige anton erna
Ich heiße  ./zeige  und habe  2 Parameter
erna anton
debian $
```

Weiterschieben der Parameter

Wenn Sie alle Parameter nacheinander bearbeiten möchten, greifen Sie zunächst mit `\$1` auf den ersten Parameter zu. Nach dessen Abarbeitung rufen Sie den Befehl `shift` auf. Dadurch wird der zweite Parameter zum ersten Parameter, der dritte zum zweiten und so weiter. Dieser Befehl wird vor allem im Zusammenhang mit Schleifen wichtig und wird darum in diesem Zusammenhang noch einmal ausführlicher behandelt.[30]

Eine Zusammenfassung der vordefinierten Variablen sehen Sie in Tabelle 4.9.

Variablen	Inhalt
`$1 $2 ..`	Parameterstrings
`$0`	Name der Skriptdatei
`$#`	Anzahl der übergebenen Parameter
`$*`	Alle übergebenen Parameter als eine Zeichenkette
`$@`	Alle übergebenen Parameter als Folge der Parameter

Tabelle 4.9 Spezielle Variablen

4.7.5 Prozessnummern

Eigene PID

In der Variablen `\$\$` findet ein Shell-Skript seine eigene Prozess-ID. Mithilfe dieser Nummer und dem Befehl `kill` könnte ein solches Skript also Selbstmord begehen. Die Kenntnis der eigenen PID hat aber auch nützliche Aspekte. So kann die PID an den Namen einer temporären Datei angehängt werden. Damit wären Überschneidungen mit parallel laufenden Prozessen ausgeschlossen. Der Dateiname */tmp/$0.$$* würde bei dem

30 siehe Abschnitt 4.8.5 Seite 144

Skript `machwas` beispielsweise den Dateinamen */tmp/machwas.2305* ergeben. Wird dasselbe Skript von einem anderen Benutzer aufgerufen, wird es eine andere PID bekommen und damit eindeutig sein.[31]

Kindprozesse

Stellt ein Skript einen Prozess in den Hintergrund, so kann es aus der Variablen `\$!` ermitteln, welche Prozess-ID der gestartete Prozess hat. Die Variable `\$?` enthält den Exit-Status des zuletzt beendeten Prozesses. Per Konvention bedeutet eine 0, dass alles in Ordnung ist. Alle anderen Zahlen repräsentieren Fehlernummern.

4.8 Ablaufsteuerung

Werden die Kommandos einfach nur hintereinander ausgeführt, spricht man von einem Batch-Lauf. Richtig interessant werden die Skripte aber, wenn sie auf äußere Umstände reagieren oder Arbeitsabläufe wiederholen können. Dazu dienen die Kommandos der Ablaufsteuerung. Zu ihnen gehören Unterscheidungen und Schleifen.

4.8.1 Die Unterscheidung: if

Abfrage

Das Kommando `if` prüft eine Bedingung und führt den hinter dem `then` angegebenen Kommandoblock nur aus, wenn die Bedingung zutrifft. Sie können optional eine andere Folge von Befehlen festlegen, die dann durchlaufen wird, wenn die Bedingung nicht zutrifft. Dieser Bereich wird durch den Befehl `else` eingeleitet. Einen `else`-Zweig müssen Sie nicht bilden. Das Kommando wird durch `fi`, also ein umgedrehtes `if`, abgeschlossen. Die Struktur einer Unterscheidung ist im folgenden Kasten dargestellt.

Struktur einer if-Anweisung

```
if <Bedingung>
then
        <Befehle>
[ else
        <Befehle> ]
fi
```

31 Das gilt natürlich nur dann, wenn die Skripte zeitgleich laufen und vor ihrem Ende ihre Dateien löschen. Auf längere Sicht kann es durchaus sein, dass nach einiger Zeit wieder ein Skript die gleiche PID bekommt.

Anwendungsfälle

Für den `if`-Befehl gibt es diverse Anwendungen. So kann ein Skript prüfen, ob ein bestimmtes Verzeichnis existiert und es anlegen, falls es noch nicht vorhanden ist. Das Skript kann ein Programm nur unter der Bedingung starten, das ein anderes Programm erfolgreich gelaufen ist.

Bedingung

Hinter dem Befehl `if` steht die Bedingung. Im einfachsten Fall ist das ein Programm. Linux bringt zwei Programme mit, die sich zum Erproben von Unterscheidungen prima eignen. Das eine Programm heißt `true`. Sein Verlauf ist immer erfolgreich. Das andere Programm heißt `false`, und es lässt sich leicht erahnen, dass dieses Programm immer ein fehlerhaftes Ergebnis liefert.

```
if true
then
   echo "gut"
else
   echo "schlecht"
fi
```

Wenn Sie diese Befehlssequenz eingeben, erscheint auf dem Bildschirm das Ergebnis »gut«. Tauschen Sie `true` gegen `false` aus, erhalten Sie die Ausgabe »schlecht«.

Hinter `then` und `else` können beliebig viele Befehle stehen. Es können sogar weitere Unterscheidungen verschachtelt werden.

Hilfe

Informationen zu `if` finden Sie normalerweise auf der Manpage der Shell, also der bash. Allerdings ist diese Manpage recht umfangreich. Schneller kommen Sie zum Ziel, wenn Sie den Befehl `help if` eingeben. Allerdings ist die Beschreibung recht kompakt.

4.8.2 Bedingungen

Für die meisten Strukturbefehle muss eine Bedingung abgefragt werden. Bei der Unterscheidung haben Sie dies ja schon gesehen. Bei Schleifen ist es nicht anders.

test

Zum Vergleich von Variablen eignet sich der Befehl `test`. Er liefert einen Wahrheitswert in Abhängigkeit von seinen Parametern. In Tabelle 4.10 sind die wichtigsten Bedingungen aufgeführt.

Ausdruck	Wirkung
`test -f` *Name*	Ist Datei *Name* eine existierende Datei?
`test -d` *Name*	Ist *Name* ein existierendes Verzeichnis?
`test` *Str*	Ist *Str* eine nichtleere Zeichenkette?
`test` *Str1* = *Str2*	Sind die Zeichenketten *Str1* und *Str2* gleich?
`test` *Str1* != *Str2*	Sind die Zeichenketten *Str1* und *Str2* ungleich?
`test` *Nr1* `-eq` *Nr2*	Ist die Zahl *Nr1* gleich *Nr2*?
`test` *Nr1* `-ne` *Nr2*	Ist die Zahl *Nr1* ungleich *Nr2*?
`test` *Nr1* `-ge` *Nr2*	Ist die Zahl *Nr1* größer oder gleich *Nr2*?
`test` *Nr1* `-gt` *Nr2*	Ist die Zahl *Nr1* größer als *Nr2*?
`test` *Nr1* `-le` *Nr2*	Ist die Zahl *Nr1* kleiner oder gleich *Nr2*?
`test` *Nr1* `-lt` *Nr2*	Ist die Zahl *Nr1* kleiner als *Nr2*?

Tabelle 4.10 Das Kommando test

[zB] Das folgende Beispielskript soll feststellen, ob es mit zwei Parametern aufgerufen wurde. Ist dies der Fall, gibt das Skript sie in umgekehrter Reihenfolge aus. Anderenfalls erscheint eine Fehlermeldung.

```
# Skript tauscht seine Parameter
if test $# -eq 2
then
    echo $2 $1
else
    echo "Falsche Parameterzahl"
fi
```

Listing 4.2 Parameterzahl prüfen

Rechteckige Klammern statt test

Da die Schreibweise mit dem Kommando `test` für Benutzer anderer Programmiersprachen sehr gewöhnungsbedürftig ist, gibt es eine alternative Schreibweise. Dabei wird das Wort `test` durch eine eckige, öffnende Klammer ersetzt, die nach dem letzten Parameter von `test` wieder geschlossen wird. Das liest sich komplizierter, als es ist. Das Beispiel macht es deutlich:

```
# Skript tauscht seine Parameter
if [ $# -eq 2 ]
then
    echo $2 $1
else
```

```
    echo "Falsche Parameterzahl"
fi
```

Listing 4.3 Rechteckige Klammer statt test

Vorsicht!

Auf eine kleine Stolperfalle muss ich Sie jedoch hinweisen: Im folgenden Skript soll eine Diskette formatiert werden, wenn der Parameter `new` angegeben wird. Das Abfragen des Parameters sieht fast genauso aus wie das Abfragen der Anzahl der Parameter.

```
# Achtung Fehler!
if [ $1 = "new" ]
then
    echo "Formatiere..."
fi
echo "Und weiter gehts..."
```

Listing 4.4 Harmlose Abfrage?

Nun rufen Sie das Skript dreimal auf. Einmal mit dem Parameter `new`, dann mit dem Parameter `old` und schließlich ohne Parameter. Und da gibt es eine Überraschung!

```
debian $ trick new
Formatiere...
Und weiter gehts...
debian $ trick old
Und weiter gehts...
debian $ trick
./trick: [: =: unary operator expected
Und weiter gehts...
debian $
```

Die Fehlermeldung sagt aus, dass der Operator = zwei Operanden erwartet, dass aber nur einer vorhanden war. Tatsächlich wird `\$1` vor dem Vergleich ausgewertet, und da die Variable keinen Inhalt hat, befindet sich zwischen der eckigen Klammer und dem Gleichheitszeichen nichts. Es ist also so, als würde dort folgender Ausdruck stehen:

```
if [  = "new" ]
```

Anführungszeichen

Sie können solche Überraschungen vermeiden, indem Sie die Variablen in Anführungszeichen setzen. Wie bereits an anderer Stelle erwähnt wurde, bewirken die Anführungszeichen das Zusammenfassen mehrerer Wörter zu einem Parameter, aber sie lassen im Gegensatz zu den Hochkommata die Auswertung der Variablen zu.

```
if [ "$1" = "new" ]
then
    echo "Formatiere..."
fi
echo "Und weiter gehts..."
```

Listing 4.5 Sichere Abfrage

Auch wenn die Variablenauswertung von `$1` leer ist, steht nun links vom Gleichheitszeichen etwas, nämlich zwei Anführungszeichen, also:

```
if [ "" = "new" ]
```

Syntax korrekt

Dieser Ausdruck ist syntaktisch korrekt. Er liefert als Wahrheitswert »falsch«, aber das soll er ja auch, da der leere String nun mal nicht einem »new« entspricht.

4.8.3 Rückgabewert von Programmen

0 ist wahr, alles andere ist falsch

Aufgerufene Programme liefern einen Wert zurück, der über den Erfolg oder Misserfolg ihrer Tätigkeit Auskunft gibt. Dabei ist es Standard, dass ein fehlerfrei gelaufenes Programm eine 0 zurückgibt. Diese wird dann von `if` als wahr interpretiert. Liefert das Programm dagegen eine Zahl ungleich 0 zurück, geht die Shell davon aus, dass es eine Fehlernummer sein wird. Darum interpretiert `if` ein Ergebnis ungleich 0 als falsch.[32]

cmp vergleicht Dateien

Einige dieser Programme sind geradezu prädestiniert für die Arbeit in Skripten. Der Befehl `cmp` vergleicht zwei Dateien. Wird er mit der Option `-s` aufgerufen, macht er dabei keine Ausgaben. Er gibt bei Gleichheit 0, anderenfalls 1 zurück.

4.8.4 Die Fallunterscheidung: case

Gestaffeltes if

Die Fallunterscheidung ist eine Art gestaffeltes `if`. Der Inhalt einer Variablen wird untersucht, und für jeden denkbaren Inhalt wird eine Aktion definiert. Eine Anwendung findet sich in den Startskripten im Verzeichnis *init.d*, die beim Booten und beim Herunterfahren des Systems gestartet werden. In diesen Skripten wird unterschieden, ob als Parameter die Wörter »start« oder »stop« übermittelt wurden bzw. ob ein anderes Schlüsselwort übergeben wurde.

---

32 Das ist für den C-Programmierer etwas ungewohnt, da dort eine 0 als falsch und alles andere als wahr interpretiert wird.

Struktur einer case-Anweisung

```
case <Variable> in
     <Maske>) <Kommandos> ;;
     <Maske>) <Kommandos> ;;
     *) <Default-Kommandos> ;;
esac
```

Eine Fallunterscheidung verzweigt in Abhängigkeit des Inhalts einer Variablen. Diese Variable steht zwischen den Schlüsselwörtern `case` und `in`. Ein Unterscheidungszweig legt zunächst den Inhalt der Variable fest, der vorliegen soll, damit dieser Zweig bearbeitet werden soll. Dieser Inhalt kann Platzhalter verwenden, wie sie bei der Namensauswahl durch die Shell üblich sind. Darum wird dieser Inhalt in der Strukturdarstellung auch als »Maske« bezeichnet. Diese wird durch eine rechte, runde Klammer abgeschlossen. Es folgen die Kommandos, die in diesem Fall ausgeführt werden sollen. Die Befehlsliste wird durch ein doppeltes Semikolon abgeschlossen. Die Fallunterscheidung endet mit dem Schlüsselwort `esac`.

[zB]

Das folgende Beispiel könnte man als Deutsch-Englisch-Wörterbuch für Arme bezeichnen. Wenn Sie mögen, können Sie es gern erweitern. Der Hauptwert dieses Skripts liegt allerdings in der Demonstration einer Fallunterscheidung.

```
case "$1" in
     Haus) echo "house" ;;
     Auto)  echo "car" ;;
     Kind)  echo "child" ;;
     *) echo "stuff" ;;
esac
```

Listing 4.6 Intelligenter Übersetzer

Default

Der Stern als letzte Maske dient zum Abfangen all der Begriffe, die durch die davor stehenden Masken nicht erfasst wurden.

Masken

Als nächstes Beispiel wird ein Skript erstellt, das `meinname` heißen und Namen analysieren soll. Der Name wird dem Skript als Parameter mitgegeben. An diesem Beispiel soll die Verwendung von Masken gezeigt werden.

```
case "$1" in
     [wW]illemer) echo "Verwandschaft!!!!" ;;
     Ar* | ar*)  echo "Welch ein Name!" ;;
```

```
    *) echo "soso! Nett, Sie kennenzulernen" ;;
esac
```

Listing 4.7 Grüß-August

Stern und rechteckige Klammern als Maske

Die erste Zeile besagt, dass der erste Parameter untersucht wird. In der zweiten Zeile sehen Sie, dass das Skript in der Lage ist, den Namen Willemer zu erkennen. Dem Skript ist es dabei egal, ob das W klein- oder großgeschrieben wird. In der nächsten Zeile werden Namen bearbeitet, die mit Ar anfangen. Der senkrechte Strich bedeutet ODER. Also ist es wieder egal, ob der Name mit einem kleinen oder großen A beginnt. Der Stern allein ist der Default, wenn keines der bisherigen Muster gegriffen hat.

4.8.5 Die while-Schleife

Wiederholungen

Schleifen ermöglichen es, Abläufe zu beschreiben, die sich wiederholen. Damit ein definiertes Ende stattfindet, läuft die Schleife nur so lange, wie eine Bedingung eingehalten wird. Diese Bedingung sollte sorgfältig gewählt werden, sonst kommt es zur gefürchteten Endlosschleife. Das würde bedeuten, dass das Programm bis zum nächsten Stromausfall läuft.

Die while-Schleife

```
while <Bedingung>
do
        <Befehle>
done
```

[zB]

Als Beispiel für eine solche Schleife sollen alle Parameter darauf überprüft werden, ob sie mit einem Minuszeichen beginnen. Dann sollen sie als Option gelten. Ansonsten handelt es sich um ein Argument. Für diese Aufgabe werden nun zwei Ablaufsteuerungen ineinander verschachtelt: eine Schleife und eine Unterscheidung. Außen läuft eine Schleife über alle Parameter. Innen findet eine Prüfung statt, ob der Parameter mit einem Minuszeichen beginnt. Spontan würde man hier `if` einsetzen, da eigentlich nur ein Abfragefall existiert. `case` hat aber den Vorteil, dass man Muster auswerten kann. Das macht die Abfrage sehr viel einfacher. Man unterscheidet einfach nach `-*` und dem Default-Fall.

```
while test -n "$1"
do
    case $1 in
        -*) echo "Option: $1" ;;
        *) echo "Argument: $1" ;;
    esac
    shift # schiebt die Parameter eine Position weiter
done
```

Listing 4.8 Optionserkennung

shift schiebt die Parameter durch

Es wird immer der erste Parameter abgefragt. In der Schleife befindet sich der Befehl `shift`. Dieser schiebt die Übergabeparameter durch. Der erste Parameter verschwindet, und alle anderen rücken eine Position nach. In der zweiten Runde ist also schon der zweite Parameter der erste geworden. Auch er wird überprüft. Und so läuft die Schleife weiter, bis es keine Parameter mehr gibt. Zur Veranschaulichung zeigt Tabelle 4.11 die Variablen `$1` bis `$5`. Jede neue Zeile zeigt die Parameter nach einem weiteren `shift`.

Anführungszeichen

In der Bedingung der Schleife ist die Variable in Anführungszeichen gesetzt. Dadurch wird das in Abschnitt 4.8.2 auf Seite 141 beschriebene Problem verhindert, wie ein Aufruf ohne Parameter zu einem Fehler führen kann. Bei `case` können die Anführungszeichen weggelassen werden, da diese Position nie erreicht wird, wenn im ersten Parameter nichts steht.

$1	$2	$3	$4	$5
anton	berta	caesar	dora	emil
berta	caesar	dora	emil	–
caesar	dora	emil	–	–
dora	emil	–	–	–
emil	–	–	–	–

Tabelle 4.11 Parameter und shift

break und continue unterbrechen eine Schleife

Schleifen können durch das Kommando `break` unterbrochen werden. Dieser Befehl steht typischerweise hinter einer `if`-Konstruktion. Allerdings kann dieser Befehl leicht zu etwas unübersichtlichem Code führen. Besser ist es, die vollständige Bedingung für das Durchlaufen einer Schleife direkt hinter dem `while` zu formulieren. In die gleiche Kategorie gehört der Befehl `continue`, der dazu führt, dass der Rest des Schleifenkörpers

nicht ausgeführt wird, sondern dass sofort zur Abfrage am Kopf gesprungen wird.

4.8.6 Die for-Schleife

Die for-Schleife ist auf die Abarbeitung von Listen spezialisiert.

Die for-Schleife

```
for <Variable> in <Liste>
do
        <Kommandos>
done
```

In der Schleife wird die Variable die Werte, die durch die Liste hinter dem Schlüsselwort `in` definiert sind, der Reihe nach annehmen. Auf die Variable kann wie üblich innerhalb der Schleife durch ein vorangestelltes Dollarzeichen (`$`) zugegriffen werden. Die Schleife wird so oft durchlaufen, wie Argumente hinter dem `in` stehen. Dabei nimmt die Variable nacheinander jedes der Argumente als Inhalt an. Beispiel:

```
for i in blau gelb grün rot
do
echo "Meine Lieblingsfarbe ist $i. Also fahre ich " \
      $i"e Autos."
done
```

Listing 4.9 Lieblingsfarben

Die Ausgabe der Schleife ist:[33]

```
Meine Lieblingsfarbe ist blau. Also fahre ich blaue Autos.
Meine Lieblingsfarbe ist gelb. Also fahre ich gelbe Autos.
Meine Lieblingsfarbe ist grün. Also fahre ich grüne Autos.
Meine Lieblingsfarbe ist rot. Also fahre ich rote Autos.
```

Variablenauflösung

Hier wird noch einmal demonstriert, dass `$i` auch innerhalb der Anführungszeichen interpretiert wird. Wollen Sie erreichen, dass der Inhalt der Zeichenkette nicht interpretiert wird, müssen Sie Hochkommata verwenden.

---

33 Nach geltender Rechtschreibung muss die Lieblingsfarbe natürlich großgeschrieben werden. Aus Bequemlichkeitsgründen lasse ich es mal so. Nach der nächsten Rechtschreibreform ist es bestimmt wieder richtig.

Wildcards

Besonders interessant wird die for-Schleife, wenn statt einer festen Liste von Zeichenketten Dateien verwendet werden, die über Wildcards ausgewählt werden. Das folgende Beispiel wandelt Audiodateien in MP3-Dateien um und löscht anschließend die Originaldateien.

```
for i in *.wav
do
  lame $i `basename $i .wav`.mp3
  rm $i
done
```

Listing 4.10 WAV nach MP3 konvertieren

basename stutzt Dateinamen

Eine kleine Schönheitsoperation wurde hier noch mit dem Kommando `basename` durchgeführt. `basename` entfernt den Verzeichnisnamen einer Datei. Wird noch ein weiterer Parameter außer dem Dateinamen angegeben, wird dieser als Anhängsel betrachtet, das von hinten abgeschnitten werden soll. Im Beispiel wird von `$i` der Anhang *.wav* abgeschnitten. An das Ergebnis dieser Operation wird *mp3* angehängt, und das Ganze wird als Zieldatei des Programms `lame` verwendet. Angenommen, im aktuellen Verzeichnis gäbe es die Dateien *a.wav*, *b.wav* und *c.wav*, dann wird die Schleife folgende Befehle erzeugen:

```
lame a.wav a.mp3
rm a.wav
lame b.wav b.mp3
rm b.wav
lame c.wav c.mp3
rm c.wav
```

4.8.7 Funktionen

Zusammenfassung von Befehlen

Sie können in einem Shell-Skript mehrere Befehle zu einer Funktion zusammenfassen. Jede Funktion erhält einen Namen, über den sie von anderer Stelle im Skript beliebig oft aufgerufen werden kann. Die Verwendung von Funktionen macht das Skript übersichtlicher und auch kürzer, da Sie Codesequenzen, die sich wiederholen, zusammenfassen können. Eine Funktion hat folgenden Aufbau:

Funktionsdefinition

<Funktionsname>()
{
    <Befehle>

```
    [ return <Rückgabewert> ]
}
```

Definition

Die Funktionsdefinition selbst wird beim Ausführen eines Skripts zunächst übersprungen. Erst wenn die Funktion aufgerufen wird, werden die darin enthaltenen Befehle ausgeführt. Ist die Funktion vollständig ausgeführt, springt der Interpreter zu der Zeile, die hinter dem Aufruf der Funktion steht. Die Funktion wird einfach durch ihren Namen aufgerufen. Hier ein sehr einfaches Beispiel:

```
meinefunktion()
{
   echo "ich tue hier etwas"
}

meinefunktion
```

Listing 4.11 Simple Funktion

Aufruf

Hier wird die Funktion namens `meinefunktion()` definiert und später einfach direkt über ihren Namen aufgerufen. Es ist also genau so, als wären alle Befehle innerhalb der Funktion an der Stelle ausgeführt worden, an der der Aufruf steht.

Parameterübergabe

Es ist auch möglich, Parameter an Funktionen zu übergeben. Der Mechanismus entspricht dem der Parameterübergabe an Skripte. Selbst der Aufruf ist identisch mit dem Aufruf eines Programms mit Parametern. Die Parameter werden einfach, durch Leerzeichen getrennt, hinter dem Funktionsaufruf aufgeführt. Innerhalb der Funktion wird auf die Parameter mit den Variablen `$1`, `$2` und so weiter zugegriffen. Listing 4.12 zeigt ein Beispiel.

```
meinefunktion()
{
    echo $1
    echo $2
}

meinefunktion "huhu"
meinefunktion "haha" 12
```

Listing 4.12 Funktion mit Parameterübergabe

Der Aufruf des Shell-Skripts bringt folgende Ausgaben auf den Bildschirm:

```
debian $ func
huhu

haha
12
debian $
```

Die Leerzeile entsteht, weil der zweite Parameter beim ersten Funktionsaufruf leer bleibt.

Rückgabe

Auch in der Skriptshell können Werte an den Aufrufer zurückgegeben werden. Dazu wird innerhalb der Funktion der Befehl `return`, gefolgt von dem Rückgabewert, eingesetzt. Der Rückgabewert kann vom Aufrufer wie der Rückgabewert eines Programms eingesetzt werden. Es gilt auch hier die Konvention, dass der Rückgabewert 0 bedeutet, dass die Funktion einwandfrei lief. Die Funktion kann also auch als boolescher Ausdruck in einer `if`-Abfrage stehen.

4.9 Ein- und Ausgaben aus dem Skript

Die einfachste Art der Ausgabe erfolgt durch den Aufruf des Befehls `echo`. Er gibt seine Parameter über die Standardausgabe aus.

Massendrucksache

Wenn Sie allerdings Texte größeren Umfangs ausgeben wollen, wie beispielsweise Hilfetexte, dann wird die Verwendung von `echo` etwas mühsam. Sie können dann dem Befehl `cat` die Eingabedatei aus dem Shell-Skript heraus geben. Und `cat` wird das tun, was es immer tut, nämlich die Datei auf der Standardausgabe ausgeben. Im Beispiel sieht das so aus:

```
cat <<!
Hier steht nun eine sehr weitschweifige Erklärung, wie das
Programm zu benutzen ist, wer der geniale Programmierer
dieser Zeilen ist und dass man nach der Benutzung dieses
Skripts nie wieder ein anderes ansehen wird.
!
```

Listing 4.13 Schwatzhaft

Der Text muss durch ein Zeichen eingeklammert werden, das im Text selbst natürlich nicht vorkommen darf. Hier ist es ein Ausrufezeichen. Der so eingegrenzte Text wird dann mit zwei Kleinerzeichen in die Standardeingabe des Befehls `cat` geschoben. Sie können auf diese Weise jedem Programm einen längeren Text in die Standardeingabe zuschieben, nicht nur `cat`.

Eingaben Hin und wieder kann es erforderlich sein, vom Anwender Eingaben zu erfragen. Dazu gibt es das Kommando `read`, das als Parameter die Variable hat, in die die Eingabe gelangen soll:

```
read ANSWER
```

Nach der Eingabe mit einem abschließenden Return wird die Eingabezeile in der Variablen `ANSWER` stehen. Im Gegensatz zu Perl gelangt das Return nicht in die Variable.

Der Befehl `read` kann auch verwendet werden, um den Eingabestrom zeilenweise zu lesen.

Sie haben Post!

22 Der Mailserver

E-Mail ist eine der ältesten Anwendungen des Internets. Die Anwendung ist dem Laien leicht verständlich. Er schreibt eine Nachricht mit einer Betreffzeile, gibt die Adresse des Empfängers an und schickt sie ab. Der Transport ist Sache des Computers und des Netzwerks.

Einfache E-Mail

Die Anwendung ist seither nicht komplizierter geworden, wohl aber die Verwaltung des Serverbetriebs. Genau damit wird sich dieses Kapitel beschäftigen. Parallel wird in Kapitel 35 ab Seite 847 ein Workshop die praktische Seite noch einmal beleuchten und begleiten.

Serverbetrieb

22.1 Übersicht und Rückblick

Bereits die alten UNIX-Maschinen in den Universitäten hatten einen Mail Transfer Agent (MTA) standardmäßig an Bord. Seinerzeit arbeiteten an einem Computer viele Teilnehmer. Eigene persönliche Computer (PCs) gab es noch nicht. Über den MTA konnten Benutzer E-Mails an andere Benutzer versenden. Jeder Benutzer besaß eine Datei, in der für ihn die Post gesammelt wurde. Auch Systemdienste nutzten diese Kommunikationsform, um den Administrator über ihre Fehlleistungen zu informieren.

Blick zurück

22.1.1 Von der lokalen Nachricht zur Internetmail

Der MTA ist auch in der Lage, E-Mails an einen anderen Computer weiterzuleiten. Das Simple Mail Transfer Protocol (SMTP) regelte den Verkehr. Die Nachrichten wurden vom Absender zum Empfänger direkt durchgeschoben. Dadurch gab es keine Verzögerung. Auf den alten Terminals erschien nach der Befehlseingabe sofort die Nachricht »Sie haben Post!«, wenn eine neue Nachricht eintraf. Benutzer einer grafischen Oberfläche erhielten von dem Programm `biff` sofort eine Meldung, wenn eine Nachricht eintraf.

MTA-Kette per SMTP

Durch die Verbreitung von persönlichen Computern, die über Modems nur zeitweise mit dem Internet verbunden waren, lag das Postfach nicht

Externe Postfächer

mehr auf dem lokalen Computer, sondern auf dem Server des Providers. Die Post musste aktiv abgeholt werden. Die Kette der per SMTP von MTA zu MTA versandten E-Mails endete beim Provider, und es musste ein neues Protokoll her, um die Post aus den Postfächern auf den lokalen PC zu holen.

POP3 und IMAP

POP3 (Post Office Protocol) ist optimal darauf ausgelegt, über zeitweise aktive Leitungen die Post vom Server abzuholen, sie lokal zu speichern und anschließend auf dem Server zu löschen. Danach kann die Verbindung wieder geschlossen werden. Im Zuge der Flatrates fand das Protokoll IMAP (Internet Message Access Protocol) Verbreitung. Hier verbleiben die Mails auf dem Server. Der Vorteil ist, dass die Mails von überall vollständig zugreifbar sind, ob vom Arbeitsplatzcomputer aus oder unterwegs per Laptop. Allerdings benötigen Sie möglichst eine Flatrate und einen Provider, der Ihnen ein paar Megabyte für Ihre E-Mails einräumt.

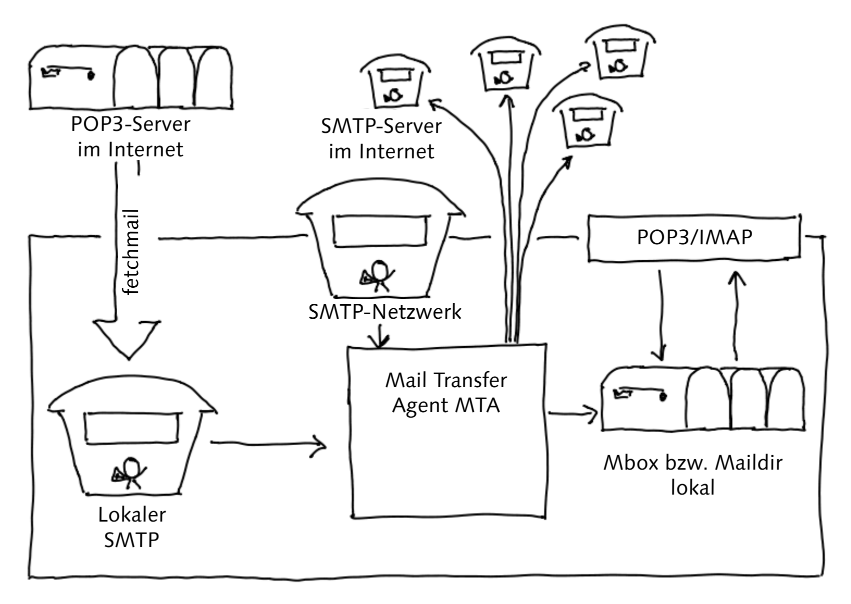


Abbildung 22.1 Übersicht über das Mailingsystem

Da das Thema recht umfangreich ist, werden die ersten Abschnitte einige Abläufe und Begriffe erläutern, bevor mit der Installation und der Konfiguration begonnen wird.

22.1.2 Vertraulichkeiten

Authentifizierung des SMTP

Die SMTP-Zugänge waren zu Anfang offen, da man sich gegenseitig half, die E-Mails zu verteilen. Ein offen zugänglicher Sendeport schien auch

unkritisch zu sein, schließlich konnte man so keine fremde E-Mail lesen. Als diese offenen Ports für das Versenden von Spam und Schadsoftware im großen Stil missbraucht wurden, musste allerdings das SMTP-Protokoll um eine Authentifizierung erweitert werden.

Klartext

E-Mails haben eigentlich eher die Merkmale einer Postkarte als die eines Briefes. Der Inhalt, der Adressat und der Absender liegen in jedem Postfach im Klartext vor, und jeder Administrator eines Mailservers zwischen Sender und Empfänger hat prinzipiell die Möglichkeit, die Nachricht zu lesen. Genauso öffentlich ist nach dem Standardprotokoll der Austausch von Benutzername und Passwort.

TLS/SSL

Für die Verschlüsselung des Nachrichteninhalts kann GnuPG verwendet werden.[1] Für die Verschlüsselung des Anmeldevorgangs müssen sich Mailserver und Mailclient einigen. Hier ist inzwischen TLS (Transport Layer Security) als Nachfolger des SSL (Secure Sockets Layer) Standard. Die Verschlüsselung erfolgt auf der Basis eines Zertifikats.

22.1.3 Massenposthaltung

MTA, POP3 und IMAP

Normalerweise verwendet das Mailsystem die lokale Benutzerverwaltung für die Mailkonten. Soll der Server große Mengen von Mailkonten verwalten, ist es nicht sinnvoll, dass jeder Mailkontoinhaber auch einen Eintrag in der lokalen Passwortdatei hat. Stattdessen können Datenbanken für die Benutzerverwaltung verwendet werden. Die Benutzerdatenbank muss sowohl vom SMTP-Server zugegriffen werden als auch von den Abholdiensten POP3 und IMAP.

Workshop

In der Anleitung in Abschnitt 35.3 ab Seite 860 wird ein Mailsystem auf der Basis einer PostgreSQL-Datenbank mit Exim MTA und Courier IMAP-Server vorgestellt.

22.2 Protokollfragen

In diesem Abschnitt werden die Protokolle näher betrachtet, die im Zusammenhang mit Mailsystemen verwendet werden. Dazu gehören die Protokolle POP3 und IMAP, die für das Lesen der E-Mails auf einem Mailserver verwendet werden. Das Protokoll SMTP ist für den MTA relevant. Im Namensdienst DNS wird festgelegt, welcher Server für die Domain zuständig ist.

1 GnuPG siehe Abschnitt 10.6.2 Seite 353

22.2.1 POP3

RFC 1939

POP3 ist wohl das gängigste Protokoll, mit dem ein E-Mail-Client die Nachrichten abholt, die auf dem Server für ihn eingegangen sind. Es ist in RFC 1939 definiert. Das POP3-Protokoll beschreibt die Authentifizierung des Benutzers und die Möglichkeiten, E-Mails aufzulisten, abzurufen und zu löschen.

Ideal für gemietete Leitungen

Dieses Verfahren ist für Telefonanbindungen ausgelegt, bei denen die Dauer der Verbindung berechnet wird. Der Client lädt seine E-Mails vom Server herunter, löscht sie dort und schließt die Verbindung. Die Bearbeitung der E-Mails erfolgt offline auf dem Arbeitsplatzrechner des Anwenders. Aber auch für den Server ist dies eine kostengünstige Lösung. Es werden nur die E-Mails gespeichert, die der Client noch nicht gelesen hat. Beim nächsten Besuch des Clients wird der Festplattenplatz in den meisten Fällen wieder freigegeben. Das Archiv der alten E-Mails führt der Anwender auf seinem Arbeitsplatz.

Kommunikation laut RFC 1939

Wie die meisten Internetprotokolle basiert auch POP3 auf dem Senden und Empfangen einfacher Textzeilen. Nach dem Kontaktieren des Servers gibt der Server sich durch eine positive Meldung als POP3-Server zu erkennen.

Serverantworten

Positive Meldungen des Servers beginnen immer mit der Zeichenfolge +OK. Es folgen oft noch weitere Angaben als Antwort auf den Befehl. Ist der Server unzufrieden, beginnt die Antwortzeile mit -ERR.

Beenden: QUIT

Die Kommunikation mit dem Server kann vom Client jederzeit mit dem Befehl `QUIT` beendet werden. Die Verbindung wird dann automatisch geschlossen.

Authentifizierung

Zu Anfang befindet sich der Server in der Authentifizierungsphase. Er erwartet, dass der Benutzer seine Legitimation beweist. Dies erfolgt meist durch Eingabe des Benutzernamens und des Passworts. Dazu wird zunächst der Befehl `USER`, gefolgt vom Benutzernamen, eingegeben. Nachdem der Server positiv bestätigt hat, dass er den Benutzer kennt, wird der Befehl `PASS`, gefolgt vom Passwort, übergeben. War auch diese Eingabe erfolgreich, wechselt der Server in die Transaktionsphase. Hier erfolgt die Bearbeitung des E-Mail-Kontos.

APOP

Alternativ gibt es noch die Möglichkeit, mit dem Befehl `APOP` eine Autorisierung durchzuführen. Dabei wird ein geheimer Schlüssel zwischen Server und Client geteilt. Der Server meldet beim Start seine Prozess-ID

und die aktuelle Uhrzeit. Der Client kombiniert sie mit dem Geheimnis und verschlüsselt sie per MD5. Diesen Wert sendet der Client dem Server als zweiten Parameter des `APOP`-Befehls.

STAT

Befindet sich die Sitzung in der Transaktionsphase, kann das Konto bearbeitet werden. Der Befehl `STAT` fordert den Server auf, seinen Zustand anzuzeigen. Der Server antwortet mit der Zeile:

```
+OK 4 44471
```

Das bedeutet, dass vier Mails vorliegen, die insgesamt 44.471 Bytes belegen.

LIST

Der Befehl `LIST` zeigt nach einer Bestätigungszeile eine Liste aller vorliegenden E-Mails an. Dabei wird für jede Mail eine Zeile ausgegeben, die nur die Mailnummer und die Größe der jeweiligen E-Mail in Byte anzeigt. Sie können dem Befehl `LIST` auch eine Mailnummer als Parameter angeben. Dann wird nur diese E-Mail angezeigt.

RETR

Mit dem Befehl `RETR` kann eine Mail vom Server ausgelesen werden. Als Argument wird eine Mailnummer erwartet. Die komplette Mail wird ausgegeben, inklusive Header und Text. Die Mail wird durch eine Zeile beendet, die nur einen Punkt enthält.

TOP

Der Befehl `TOP` ist nach RFC 1939 nicht zwingend vom Server zu implementieren. In der Praxis dürfte er aber immer vorhanden sein. Mit diesem Befehl können die ersten Zeilen einer Mail geholt werden. Es werden in jedem Fall der Header und ein paar Zeilen der eigentlichen Nachricht geladen. Der TOP-Befehl muss die Mailnummer als Parameter übergeben und kann zusätzlich die Anzahl der Zeilen bestimmen, die er von der Nachricht lesen will.

DELE

Wenn eine Mail auf dem Server gelöscht werden soll, verwenden Sie den Befehl `DELE`, gefolgt von der Mailnummer. Damit wird die Mail als gelöscht markiert und von allen Befehlen der Sitzung so behandelt, als sei sie nicht vorhanden. Erst wenn die Sitzung in die `UPDATE`-Phase kommt, wird die Nachricht endgültig gelöscht.

RSET

Der Befehl `RSET` hebt alle durch `DELE` gesetzten Markierungen auf.

NOOP

Mit `NOOP` deutet der Client an, dass er momentan nichts tut, aber den Server immer noch lieb hat. Auf diese Anweisung reagiert der Server mit einer positiven Bestätigung. Dieser Befehl kann nützlich sein, um beispielsweise einen Timeout zu verhindern.

Update-Phase

Wird nach der Transaktionsphase der Befehl `QUIT` gesendet, geht der Server in die Update-Phase und löscht alle mit einer Löschmarke versehenen Mails.

Eine kleine Beispielsitzung

Protokoll per telnet

Sie können eine POP3-Sitzung mithilfe des Programms `telnet` leicht sichtbar machen oder prüfen. Im Folgenden wurde eine kleine Sitzung protokolliert. Sie besteht aus der Anmeldung, der Anzeige, wie viele Mails vorliegen, aus dem Herunterladen einer E-Mail und aus dem Ende der Sitzung. Die eigenen Eingaben sind eingerückt.

```
squeeze $ telnet debian 110
Trying 192.168.109.199...
Connected to debian.
Escape character is '^]'.
+OK ready  <3745.1014213784@debian.willemer.edu>
   User andrea
+OK Password required for andrea.
   pass daswerdeichhierauchgeradeimklartextschreiben
+OK andrea has 4 visible messages (0 hidden) in 44471 octets.
   LIST
+OK 4 visible messages (44471 octets)
1 12980
2 4064
3 3673
4 23754
.
   RETR 1
+OK 12980 octets
Return-Path: <ems+HA564Q8DYULUE6@bounces.amazon.com>
Received: from localhost (localhost [127.0.0.1])
 ...
Mit vorzüglicher Selbstbeherrschung
Roswita Presswurst

.
   QUIT
+OK Pop server at debian.willemer.edu signing off.
Connection closed by foreign host.
squeeze $
```

Eine Konversation

Die eingerückten Zeilen wurden als Kommandos in `telnet` direkt eingegeben. Im normalen Betrieb übermittelt diese der POP3-Client. Das erste Kommando, `User andrea`, meldet den Benutzer an. Der Benutzername wird vom Server aus */etc/passwd* entnommen. Als Nächstes wird das Pass-

wort gesendet. Der Befehl `pass` leitet es ein. Der Server bestätigt mit OK die korrekte Anmeldung und gibt an, dass vier Nachrichten vorliegen, die zusammen 44.471 Bytes[2] belegen. Mit dem Befehl `LIST` erhält der Client eine Liste von vier Nachrichten mit deren Größen. Der einzelne Punkt schließt die Liste ab. Der Befehl `RETR 1` holt die erste Nachricht. Sie erscheint im Klartext, beginnt mit dem Header und endet mit einem einzelnen Punkt. Zuletzt wird die Sitzung durch das Kommando `QUIT` beendet.

22.2.2 IMAP

E-Mails bleiben auf dem Server

IMAP4 ist in RFC 1730 definiert, IMAP4r1 in RFC 2060. Die aktuelle Version ist in der RFC3501 beschrieben. IMAP (Internet Message Access Protocol) bezeichnet ein Protokoll zur Verwaltung von E-Mails. Im Gegensatz zu POP3 verbleiben die Mails auf dem Server und werden dort verwaltet. So ist es möglich, die Mail von mehreren Clients aus zu verwalten, da die Nachrichten nicht vom Server gelöscht werden. Die Mails können auf dem Server in Ordnern einsortiert werden, sodass der Unterschied zu einem lokalen POP3-Client nur gering ist. Der Hauptunterschied liegt darin, dass Sie zur Bearbeitung Ihrer Mails online sein müssen und dass der Server Ihnen genügend Speicher anbieten muss. Auch das Thema Datensicherung vereinfacht sich, weil die Mails zentral liegen.

Textorientiert

Ähnlich wie POP3 ist auch IMAP ein textorientiertes Protokoll, das mit dem Server Befehle austauscht. Mit dem Befehl `LOGIN` meldet sich der Client an. Mit `SELECT` wählt er den Arbeitsordner. Der Eingangsordner für neue Mails heißt *INBOX*. Der Server liefert mehrere Zeilen. Er gibt an, wieviele Mails existieren und welche der Mails bisher noch nicht gelesen wurde. Mit dem Kommando `FETCH` kann eine einzelne Mail geladen werden. Mit dem Befehl `STORE` werden schreibende Kommandos an den Server gesendet, beispielsweise zum Löschen der Nachricht auf dem Server.

Definition

Da auch Befehle zum Verwalten der Ordner existieren, ist der Umfang etwas größer als bei POP3. Sie finden eine ausführliche Beschreibung des Protokolls unter folgender URL:

http://tools.ietf.org/html/rfc3501

---

2 Oktett ist der von humanistisch gebildeten Informatikern präferierte Ausdruck für Byte, das bekanntermaßen aus acht Bit besteht. Das Wort leitet sich vom dänischen Wort »otte« für »acht« her. König Ottokar von Dänemark, genannt der Unachtsame, hatte nämlich acht Kinder.

22.2.3 SMTP

RFC 821 RFC 1869

Das Simple Mail Transport Protocol (SMTP) ist für den Empfang und die Verteilung von Mails zuständig. Es wurde 1982 unter der RFC 821 beschrieben. Das Protokoll wurde 1995 als Extended SMTP (ESMTP) in RFC 1869 erweitert. Wichtige Themen sind vor allem die Authentifizierung der Clients und die modulare Erweiterung des Servers.

Binärdaten

SMTP ist ein reines ASCII-Protokoll mit allen Befehlen in Klartext. Die Übertragung von Binärdaten war ursprünglich nicht vorgesehen. Der MIME-Standard ermöglicht die Umcodierung der binären Daten im Anhang als reine Textzeichen, sodass der Mailserver gar nicht merkt, dass er Binärdaten überträgt. Der Nachteil des Verfahrens ist, dass angehängte Dateien für den Transport deutlich größer werden.

Protokollablauf

Eine SMTP-Sitzung beginnt damit, dass sich der Server mit einem Gruß meldet und darauf wartet, dass sich der Client authentifiziert. Jede Meldung des Servers beginnt mit einer Nummer, die einen Hinweis darauf gibt, wie der letzte Befehl verarbeitet wurde.

- **1xx**
  Der Befehl wurde akzeptiert, aber der Server wird noch nicht tätig, sondern erwartet eine Bestätigung.
- **2xx**
  Der Befehl wurde akzeptiert und ausgeführt.
- **3xx**
  Der Befehl wurde verstanden, es werden aber noch weitere Informationen vor der Verarbeitung benötigt.
- **4xx**
  Es ist ein temporärer Fehler aufgetreten. Eine Wiederholung des Befehls könnte erfolgreich sein.
- **5xx**
  Es ist ein ernster Fehler aufgetreten. Der Befehl wird nicht ausgeführt.

Der Client gibt Befehle, die aus vier Buchstaben bestehen. Es werden in der Regel direkt hinter dem Befehl weitere Parameter übergeben.

Hallihallo

Mit dem Befehl `HELO` eröffnet der Client die Sitzung. Als Parameter gibt er seinen Hostnamen inklusive Domäne an. Wird der Befehl `EHLO` statt `HELO` gesendet, gibt der Client an, dass er das erweiterte SMTP beherrscht, und der Server meldet seine Erweiterungen. Hier ein kurzer Protokollmitschnitt:

```
220 debian.willemer.edu ESMTP Postfix (Debian/GNU)
EHLO squeeze.willemer.edu
250-debian.willemer.edu
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
```

Übertragen einer Mail

Mit dem Befehl `MAIL FROM:` gibt der Client den Absender der Nachricht an. Es folgt die Mail-Adresse des Absenders. Der Befehl `RCPT TO:` gibt den Empfänger an, dessen Mailadresse wieder als Parameter angehängt wird. Mit dem Befehl `DATA` wird dann der Mailinhalt Zeile für Zeile übertragen. Der Befehl ist abgeschlossen, wenn eine Zeile gesendet wird, die nur einen Punkt enthält. Der Inhalt einer Nachricht besteht aus dem Header, der Absender, Empfänger, Datum, Uhrzeit und Zwischenstationen enthält, und dem eigentlichen Textkörper. Der Übergang wird durch eine leere Zeile angezeigt.

Und Tschüß

Mit dem Befehl `QUIT` wird die Verbindung zwischen Client und Server regulär geschlossen.

22.2.4 Mailserver und Domain

Domäne

In einer E-Mail-Adresse befindet sich links neben dem @-Zeichen der Benutzername und rechts der Hostname des Computers mit seiner Domain. Welcher Rechner innerhalb einer Domain die Mails entgegennimmt, interessiert den Versender der Mail eigentlich nicht. So richten sich Mails, die von außen kommen, normalerweise nur an die Domain. Damit der weiterleitende SMTP-Server aber weiß, welcher Computer für den Empfang der Mails der Domäne zuständig ist, gibt es beim DNS (Domain Name Service[3]) einen besonderen Eintrag für den Mailserver, der mit `MX` gekennzeichnet ist.

```
@       IN SOA    mail.willemer.edu.  root.mail.willemer.edu.
                  (
                   ......
                  )
;                 Wer sind die zustaendigen Mailserver
                  IN MX 10 mail.willemer.edu.
```

3 DNS siehe Kapitel 21 Seite 651

MX-Eintrag

In diesem Beispiel ist der Computer namens *mail* für die Domäne *willemer.edu* zuständig. Die Zahl hinter `MX` ist die Priorität. Werden mehrere Server aufgelistet, wird zuerst derjenige mit der kleinsten Nummer ausgewählt. Erst wenn dieser nicht ansprechbar ist, wird die Post an denjenigen mit der nächstkleineren Nummer gesendet. Dieser Server wird in regelmäßigen Abständen versuchen, die Mail an den primär zuständigen Mailserver weiterzugeben, auf dem die Anwender ihre Mails abholen. Nähere Details zum DNS-Eintrag finden Sie in Abschnitt 21.2 ab Seite 659.

22.3 Die Erbschaft der UNIX-Mail

UNIX als Urvater von Linux hatte bereits auf jedem System ein voll funktionsfähiges Mailsystem an Bord. Auch wenn nicht mehr alle Bestandteile dieses Mailsystems heute noch gebräuchlich sind, so hat dieses System doch Standards gesetzt, die heute noch in Gebrauch sind.

22.3.1 Uralt-Client mail

Client mail

Von jedem Terminal aus können Sie eine Mail absenden. Sie geben `mail` gefolgt von dem Adressaten, an. Das muss keine vollwertige Mail-Adresse sein, sondern es reicht der Benutzername eines lokal eingerichteten Kontos. Anschließend erscheint die Eingabemöglichkeit für einen Betreff (Subject). Anschließend können Sie Zeile für Zeile Text eingeben. Als Abschluss geben Sie eine leere Zeile ein, die nur einen Punkt enthält. Anschließend wird die Mail dem lokalen MTA (Mail Transport Agent) zur Weitervermittlung übergeben.

Mail abrufen

Wenn Sie `mail` ohne Adressaten eingeben, können Sie Ihre eingegangenen Mails einsehen und gegebenenfalls löschen. Das Programm ist sehr simpel, aber für kleine Experimente durchaus brauchbar.

MTA sendmail

Der MTA ist für die Verteilung der Mails zuständig. Der Standard-MTA auf UNIX-Maschinen und damit auch auf Linux-Computern hieß lange Zeit sendmail. Das Programm ist sehr leistungsfähig, aber gefürchtet für seine umfangreiche und leicht kryptische Konfiguration. Tatsächlich finden Sie auf fast jedem System immer noch den Befehl `sendmail` im Verzeichnis */usr/sbin*. Allerdings handelt es sich nicht um ein Stück prähistorische Software, der man einen Altersruhesitz gegönnt hat, sondern um einen Verweis auf den jeweiligen MTA. So ist *sendmail* bei Exim4 ein symbolischer Link auf die Programmdatei *exim4*.

```
debian # ls -l /usr/sbin/sendmail
lrwxrwxrwx 1 root root 5  5. ... /usr/sbin/sendmail -> exim4
```

22.3.2 Mailablage Mbox oder Maildir

Eingelagerte Mbox

Standardmäßig werden Mails im Verzeichnis */var/mail* gespeichert. Dort wird für jeden Benutzer eine Datei angelegt. Neue Mails werden einfach an diese Datei angehängt. Diese Datei ist nicht verschlüsselt, aber gegen fremdes Lesen durch seine Dateirechte geschützt.

Schönheitsfehler

Das Verfahren, die Mails im Mbox-Verfahren abzulegen, hat ein paar Schönheitsfehler. Das erste Problem stellt die Tatsache dar, dass alle Mails in einer großen Textdatei enthalten sind. Schon die Größe stellt ein Problem dar, da heute Grafiken, Bilder und Filme im Umfang mehrerer MByte als Anhang versandt werden. Entsprechend groß kann die Mbox-Datei werden, und das Suchen nach einzelnen Mails gestaltet sich langwierig. Mails werden immer wieder gelöscht. Es müssen also Bereiche »herausgeschnitten« werden. Dazu müssen große Bereiche der Datei verschoben werden. Das wird umso aufwendiger, je größer die Datei ist. Das zweite Problem liegt darin, dass die Dateien im Verzeichnis */var/mail* liegen. Damit liegen diese Daten immer im Verantwortungsbereich des Administrators. Würden sie im Benutzerverzeichnis liegen, hätte der Benutzer die volle Kontrolle und die volle Verantwortung. Und bei der Datensicherung muss das Mbox-Verzeichnis nicht explizit mitbedacht werden.

Alternative Maildir

Die Alternative heißt Maildir. Die Mails werden in einem speziellen Verzeichnis abgelegt, das im Benutzerverzeichnis liegt. Unterhalb dieses Verzeichnisses, dass meist *Maildir* heißt, liegen weitere Verzeichnisse, die zur Strukturierung der Mails dienen. Für jede Mail wird eine eigene Datei angelegt.

mail auf Maildir umstellen

mail und Maildir

Einige ältere Programme verwenden standardmäßig noch das Mbox-Format. Aber sie können meist auch anders. Beispielsweise kann man sogar das Programm `mail` so konfigurieren, dass es mit dem Maildir-Format klarkommt. Es müssen nur in zwei Dateien Änderungen vorgenommen werden: */etc/mail.rc* und */etc/profile*.

```
set folder="Maildir/"
set MBOX="Maildir"
set record="~/Maildir/sent-mail/"
```

Listing 22.1 /etc/mail.rc

Umgebung

In der Datei */etc/profile* werden einige Umgebungsvariablen vorbesetzt.

```
export MAIL="/home/$USER/Maildir"
export MAILDIR="~/Maildir"
```

Listing 22.2 /etc/profile

Umstellung auf Maildir

mb2md

Eine Umwandlung der vorliegenden Mails kann durch das Programm mb2md erreicht werden. Dieses muss allerdings unter Debian explizit installiert werden.

```
debian # apt-get install mb2md
```

Mit dem folgenden Aufruf sortieren Sie für den Benutzer *georg* die Mbox-Datei in das Maildir-Verzeichnis um:

```
debian # mb2md -s /var/mail/georg -d /home/georg/Maildir
```

Vorsicht!

Als ganz verlässlich hat sich das Skript allerdings nicht erwiesen. Prüfen Sie also zunächst die Ergebnisse, bevor Sie sich von den Original-Mbox-Dateien trennen.

22.3.3 Benutzerzuordnung mit aliases

/etc/aliases

Die Datei */etc/aliases* ist der Verschiebebahnhof für die Mails zwischen den Benutzern. Beispielsweise nutzen Mails an root nichts, wenn root diese niemals liest. Es wäre besser, die Nachrichten an den Zivilbenutzer zu senden, den root für die Normalarbeiten verwendet.

```
root : johannes
```

newaliases

Nun werden alle Nachrichten an den Benutzer *johannes* weitergeleitet. Allerdings wird das erst passieren, wenn Sie einmal nach der Änderung der *aliases*-Datei den Befehl newaliases aufrufen, der dann die Inhalte der Datei *aliases* in die Datei *aliases.db* übernimmt.

Wenn Sie es genau nehmen, gehört die Aliasdatei zu dem Paket *sendmail*. Allerdings übernehmen auch andere Mail Transport Agents aus Kompatibilitätsgründen diesen Mechanismus.

22.4 Standard MTA Exim

Debian-Standard

Ein Linux-System bringt immer einen Mailserver mit, weil einige interne Prozesse ihr Unbehagen über unangenehme Ereignisse in Mails an den

Administrator zusammenfassen. Debian verwendet Exim4 in der Light-Version als Standardmailsystem. Exim4 ist in der Standardkonfiguration bereits in der Lage, eintreffende Nachrichten an die Benutzer des Systems weiterzuleiten.

Installation

Sollten Sie Exim4-Light aus irgendeinem Grund neu installieren müssen, verwenden Sie dafür den folgenden Befehl:

```
debian # apt-get install exim4
```

Vollversion

Wenn Sie keine Gewichtsprobleme mit Ihrem Rechner haben, können Sie die »Heavy-Version« einsetzen. Das Paket heißt *exim4-daemon-heavy* und wird vor allem dann benötigt, wenn eine der folgende Eigenschaften eingesetzt werden soll:

- LDAP-Unterstützung
- Datenzugriffe auf PostgreSQL- oder MySQL-Datenbanken
- SASL- und SPA-SMTP-Authentifizierung
- Eingebetteter Perl-Interpreter
- exiscan-acl für die Integration des SpamAssassin oder Virenscanner

Installiert wird das Paket mit dem Aufruf:

```
debian # apt-get install exim4-daemon-heavy
```

22.4.1 Mitgelieferte Informationen

/usr/share/doc

Im Verzeichnis */usr/share/doc/exim4-base* finden Sie eine Datei namens *README.Debian.gz*, die eine ausführliche Konfigurationsbeschreibung enthält. Sie können sie mit dem Programm `zless` lesen.

```
debian $ cd /usr/share/doc/exim4-base
debian $ zless  README.Debian.gz
```

HTML-Version

Dieser Text liegt unter dem Namen *README.Debian.html* auch im HTML-Format vor. Sie finden ihn im gleichen Verzeichnis. Sie können also alternativ auch einen Browser zum Lesen verwenden.

spec.txt

In diesem Verzeichnis finden Sie auch eine Datei namens *spec.txt.gz*, die Sie ebenfalls mit `zless` auslesen können. Sie enthält die Spezifikation von Exim4 und ist zwar entsprechend umfangreicher, aber formaler und damit nicht so flüssig zu lesen.

Konfigurationsdateien

Für die Konfigurationsdateien finden Sie ausführliche Manpages in Abschnitt 5. Sie rufen sie mit folgendem Befehl auf:

```
debian $ man 5 exim4-config_files
```

Hier finden Sie eine Übersicht über alle Konfigurationsdateien, die Exim betreffen, mit einer kurzen Beschreibung, welche Aufgaben sie haben.

22.4.2 Grundkonfiguration

dpkg-reconfigure

Sie können unter Debian das Tool `dpkg-reconfigure` für die Konfiguration verwenden. Dieses erstellt im Dialog mit dem Administrator die wichtigsten Einstellungen.

```
debian # dpkg-reconfigure exim4-config
```

Konfigurationsdatei

Die Eingaben in den nun folgenden Dialogen führen in den meisten Fällen zu einer Veränderung der Datei */etc/exim4/update-exim4.conf.conf*.

Beispielumgebung

Für die Beispielumgebung wird die eines freiberuflichen Buchautors unbekannter Herkunft verwendet. Der lokale Mailserver heißt *debian* und hat die IP-Adresse 192.168.109.199. Die Subnetzmaske lautet 255.255.255.0. Die lokale Domäne heißt *willemer.edu*, weil gewährleistet ist, dass dieser Name im Internet nicht vergeben wird. Die Domäne *willemer.de*, die auch als Absender bei den Mails verwendet wird, wird von einem Domänenprovider verwaltet, der als Smarthost dienen soll. Sie werden natürlich die Angaben an Ihre Bedürfnisse anpassen müssen.

Aufgabenstellung

Gleich zu Anfang fragt das Programm nach dem grundsätzlichen Einsatzbereich des Mailservers. Es werden folgende Möglichkeiten angeboten:

- **internet**
  Internetserver: E-Mails werden direkt an den SMTP-Server der Zieldomäne verschickt. Mails an die eigene Domäne werden empfangen und direkt in die Mailboxen der User verteilt.

  Diese Option ist für große Firmen oder Provider interessant, die ihren Mailserver mit direkter Verbindung zum Internet positionieren.

- **smarthost**
  Versand über Sendezentrale (Smarthost); Empfang mit SMTP oder Fetchmail. Exim4 verteilt die lokalen Nachrichten selbst und leitet die Mails ins Internet an einen Smarthost weiter. Ein solcher Smarthost kann beispielsweise der Mailserver Ihrer Domäne sein.

- **satellite**
  Versand über Sendezentrale (Smarthost); keine lokale E-Mail-Zustellung. Diese Einstellung eignet sich für Rechner, die als Arbeitsplatzrechner in einem Netzwerk stehen.

- **local**
  Nur lokale E-Mail-Zustellung; keine Netzwerkverbindung
- **none**
  Keine Festlegung zum jetzigen Zeitpunkt. Etwaige vorige Konfigurationen bleiben erhalten.

Smarthost

Im Folgenden wird davon ausgegangen, dass die Wahl auf den zweiten Punkt fiel. Auf diese Weise werden lokale Mails verteilt und Mails für das Internet an einen freundlichen Dienstleister übergeben.

Je nach Antwort des Benutzers füllt das Programm die Variable `dc_eximconfig_configtype` in der Datei *update-exim4.conf.conf*. Beispielsweise würde für die Antwort des Versands über die Sendezentrale das Wort »smarthost« eingetragen.

```
dc_eximconfig_configtype='smarthost'
```

E-Mail-Name des Systems

Mail-Domäne

Das Programm bittet Sie um die Eingabe eines E-Mail-Domänennamen des lokalen Computers. Dieser Name ergänzt die lokalen Benutzernamen zu einer vollständigen Mail-Adresse. So wird eine lokale Mail an den Benutzer *georg* zur gültigen E-Mail-Adresse georg@debian.willemer.edu. Andererseits wird der Server eine Mail an georg@debian.willemer.edu automatisch dem lokalen Benutzer *georg* zuordnen.

```
dc_other_hostnames='debian.willemer.edu'
```

Betreute Netzwerkschnittstellen

IP-Adressen eingehender Verbindungen

Der Server kann prinzipiell von allen Netzwerkschnittstellen aus angesprochen werden. In der nachfolgenden Eingabe können Sie die IP-Adressen der Schnittstellen angeben, die der Server bedient. Wenn Sie diesen Eintrag leer lassen, werden von allen Netzwerkadressen SMTP-Anfragen akzeptiert. Soll das Einstellen von Mails nur auf dem lokalen Rechner für die lokale Post erlaubt sein, tragen Sie die localhost-Adresse 127.0.0.1 für IPv4 oder ::1 für IPv6 ein. Sie können auch beides eintragen. Dazu setzen Sie einfach ein Semikolon zwischen die Einträge.

Weitere Schnittstellen

Soll auch die Ethernetschnittstelle des lokalen Rechners bedient werden, geben Sie die IP-Adresse der lokalen Ethernetkarte ein. In der Regel erfahren Sie diese mit dem Befehl `ifconfig eth0`.

```
debian # ifconfig eth0
eth0  Link encap:Ethernet Hardware Adresse 00:25:22:1e:db:d8
      inet Adresse:192.168.109.199  Bcast:192.168.109.255...
      inet6-Adresse: fe80::225:22ff:fe1e:dbd8/64 Gültigke...
```

IP-Adresse

Auf dem Beispielsystem ist es die Adresse 192.168.109.199. Erst wenn diese IP-Adresse zusätzlich zur Vorgabe von localhost eingetragen ist, wird eine Anfrage aus dem Netzwerk Beachtung finden. Der Eintrag, den Sie im Dialog eingeben, wird in der Variablen `dc\_local\_interfaces` hinterlegt.

```
dc_local_interfaces='127.0.0.1 ; ::1 ; 192.168.109.199 '
```

Es ist besser, Sie führen alle Schnittstellen auf, obwohl es einfacher wäre, das Feld einfach leer zu lassen. So werden Sie sich später nicht wundern, wenn der Server eine kurzfristige Modemverbindung nutzt, um auch im Internet zur Verfügung zu stehen.

Empfängerdomänen

E-Mail-Adresse

Im nächsten Schritt wird bestimmt, welche E-Mail-Adressen lokal verteilt werden und nicht an das Internet weitergeleitet werden sollen. Man kann diese Domänen auch als diejenigen bezeichnen, für die der Rechner das endgültige Ziel sein soll. Der Hostname und localhost werden automatisch vorgegeben. Hier können Sie noch weitere Domänennamen hinzufügen.

```
willemer.edu
```

In der Konfigurationsdatei wird der Inhalt in die Variable `dc_readhost` geschrieben.

```
dc_readhost='willemer.edu'
```

Relay

Domäne

Nun werden die Domänen erfragt, für die dieser Server einen Relay bilden soll. Das heißt, dass die Mails für diese Domänen an den Smarthost weitergeleitet werden sollen. Dazu sollte dieser Server in dem MX-Eintrag des DNS-Servers dieser Domäne eingetragen sein.

```
dc_relay_domains='willemer.edu'
```

Weiterleitende IP-Bereiche

Nun wird bestimmt, welche IP-Adressen in diesem Server ihre an den Smarthost ausgehenden E-Mails ablegen dürfen. Soll der Server zur Weiterleitung der Mails des lokalen Netzwerks dienen, geben Sie hier die IP-Adresse des Netzwerks ein, beispielsweise 192.168.109.0/24.

```
dc_relay_nets='192.168.109.0/24'
```

Vorsicht!

Bitte denken Sie daran, dass alle Netze, die Sie hier eintragen, freie Übertragungsrechte besitzen. Wenn Sie später eine Anmeldung per Passwort für diesen Mailserver vorschreiben, wird dieses nicht für die hier eingetragenen Netzwerke gelten.

Smarthost

Versendungshilfe

Sie können einen fremden Rechner angeben, der als ausgehende Sendezentrale verwendet werden soll. Man spricht in diesem Fall von einem Smarthost. Hier steht beispielsweise der SMTP-Server Ihres Providers. Den brauchen Sie, wenn Ihr Mailserver nicht selbst mit einer festen IP-Adresse im Internet steht. Typische Eingaben sind *smtp.1und1.de* oder *mail.gmx.de*. Sie können den Smarthost direkt in der Konfigurationsdatei eintragen.

```
dc_smarthost='smtp.1und1.de'
```

Authentifizierung

Sofern der Smarthost nicht ein Rechner ist, der zum lokalen Netzwerk gehört und seinen Kollegen vertraut, werden Sie bei jedem Anbieter darauf stoßen, dass er eine Authentifizierung des Benutzers fordert. Dazu muss die URL des Smarthost, die Benutzerkennung und das Passwort in der Datei *passwd.client* im Verzeichnis */etc/exim4* hinterlegt werden. Dort geben Sie jeweils, durch einen Doppelpunkt voneinander getrennt, den Smarthost, den Benutzernamen und das Passwort an.

```
# /etc/exim4/passwd.client
smtp.1und1.de:georg:supergeheim
```

Verbergen der lokalen Domäne

Wegblenden

Wenn Sie lokal eine nicht im Internet angemeldete Domäne verwenden, ist es notwendig, dass diese in den versendeten E-Mails verborgen wird und durch die Maildomäne ersetzt wird, über die Sie auch die Antworten erwarten. Dann geben Sie hier »Ja« an. Geben Sie »Nein« ein, wenn die lokal verwendete Mail-Adresse auch Ihrer Internetadresse entspricht.

```
dc_hide_mailname='true'
```

Wenn Sie die Adresse verbergen, werden Sie gefragt, wie die Domäne heißen soll, unter der die E-Mail versandt werden soll.

DNS-Anfragen minimieren

Zugangsfragen

Wenn Sie noch eine ISDN- oder Modemverbindung verwenden, müssen Sie jede Anfrage an einen DNS bezahlen. Insofern können Sie hier die

Anfragen minimieren. Verwenden Sie eine Flatrate brauchen Sie keine Minimierung.

```
dc_minimaldns='false'
```

Mbox oder Maildir

Der klassische Standard für das Ablegen der Mails in UNIX-Systemen wird Mbox genannt. Dabei wird für jeden Benutzer eine Datei mit dessen Nutzernamen im Verzeichnis */var/mail* abgelegt. Insbesondere für das Zusammenspiel mit dem Courier POP3- und IMAP-Server ist die Maildir-Struktur sinnvoller.

```
dc_localdelivery='maildir_home'
dc_localdelivery='mail_spool'
```

Einstellungen auf kleine Dateien

Große Datei

Exim kann auf zwei Arten seine Konfigurationen verwalten. Sie können eine große Datei */etc/exim4/exim4.conf* verwenden. Oder Sie können die Konfiguration auf viele kleine Dateien verteilen lassen. Diese befinden sich in Unterverzeichnissen des Verzeichnisses */etc/exim4/conf.d/*. Für die folgende Konfiguration ist hier »Nein« angewählt worden.

Neustart

Wenn Sie die Konfiguration durchgeführt haben, wird der Exim-Server automatisch neu gestartet. Sie können den Exim-Server jederzeit mit dem folgenden Befehl über das Init-Skript neu starten:

```
debian # /etc/init.d/exim4 start
```

Workshop

Mit der Einstellung dieser Grundkonfigurationen können Sie Exim bereits für den Mailbetrieb innerhalb des Rechners, des lokalen Netzwerks oder des Internets einrichten. Im Workshop in Abschnitt 35 ab Seite 847 finden Sie ein paar Beispielszenarien, die den Umgang mit Exim und das Zusammenspiel mit anderen Komponenten beschreiben.

22.4.3 Fehlerprotokolle

Für die Fehlerprotokolle hat sich Exim ein eigenes Verzeichnis *exim4* unterhalb des normalen Protokollverzeichnisses */var/log* gegönnt. Die meisten Fehler und Informationen werden in die Datei *mainlog* protokolliert. Zurückgewiesene Mails werden in der Datei *rejectlog* vermerkt. Täglich wird die Dateien wegrotiert und komprimiert, sofern neue Meldungen eingetroffen sind.

Panische Reaktion

Wenn Sie eine Konfiguration von Exim erstellen, die heftige Fehler aufweist, dann kann es passieren, dass in die Fehlerdatei *paniclog* eine Meldung geschrieben wird. Diese Datei findet sich im Verzeichnis

/var/log/exim4. Solange diese Meldung nicht von Hand entfernt wird, erhalten Sie beim Neustart immer wieder folgende Meldung:

```
Restarting MTA: exim4.
ALERT: exim paniclog /var/log/exim4/paniclog has
non-zero size, mail system possibly broken ... failed!
```

Löschen!

Es kann etwas irritierend sein, weil Sie eventuell den zugehörigen Fehler längst entfernt haben. Da die Fehlermeldung aber immer noch in dieser Datei steht, wird die Meldung bleiben und Exim nicht starten. Die Abhilfe ist ganz einfach: Editieren Sie die Datei, und löschen Sie die Meldung, oder verwenden Sie den folgenden Befehl:

```
debian # > /var/log/exim4/paniclog
```

22.4.4 Konfigurationsdateien und Makros

Makros als Flags

Für die Konfiguration von Exim werden oft Makros eingesetzt. Diese bezeichnen eine besondere Aufgabenstellung. In den bereits vorbereiteten Szenarien der Konfigurationsdateien werden diese Makros abgefragt. Ein typisches Beispiel ist das folgende Makro, das eine TLS-Authentifizierung auslöst:

```
MAIN_TLS_ENABLE = true
```

TLS-Aktivierung

An verschiedenen Stellen der Konfigurationsdatei wird dieses Makro geprüft, und es werden besondere Schritte hinzugefügt, die nur bei einer TLS-Übertragung erforderlich sind. So wird die TLS-Übertragung nur einmal konfiguriert und führt nicht dazu, dass an verschiedenen Stellen Änderungen gemacht werden müssen.

Definition

Makros müssen vor ihrer ersten Auswertung definiert werden. Sie können sie entweder in der Datei *exim4.conf.template* gleich zu Anfang setzen, oder Sie legen eine Datei *exim4.conf.localmacros* an, in der diese Definitionen erfolgen.

Verteilte Konfiguration

Sollte eine Konfiguration mit aufgeteilten Konfigurationsdateien verwendet werden, gehören die Makros in die Datei */etc/exim4/conf.d/main* oder in eine Datei, die verlässlich zuerst gelesen wird, beispielsweise *000_localmacros*.

22.4.5 Verschlüsselt zur Post

TLS

Der Transport von Passwörtern wird heutzutage verschlüsselt durchgeführt. Und auch der Inhalt der Mails, die zur Post gebracht werden, müssen ja nicht jedem im Internetcafé auf die Nase gebunden werden. Das Zauberwort heißt TLS (Transport Layer Security) und sichert die Partner vor Mithörern. Eine TSL-Verschlüsselung für Mailserver basiert auf einem Zertifikat, das von einer autorisierten Stelle signiert ist. Ein unsigniertes Zertifikat ist in lokalen Umgebungen immerhin besser als gar keins.

Makrodefinition

Um die Verschlüsselung in Exim einzuschalten, wird in der Datei */etc/exim4/exim4.conf.template* die Variable MAIN_TLS_ENABLE gesetzt. Dadurch wird es ermöglicht, über Port 25 eine TSL-Anmeldung durchzuführen. Allerdings erwarten viele Clients für den TSL-gesicherten Server den Port 465, wie er auch in der Datei */etc/services* als ssmtp aufgeführt ist. Dazu wird die Variable `tls_on_connect_ports` entsprechend gesetzt. Beides passiert in der Datei *exim4.conf.templates*.

```
# /etc/exim4/exim4.conf.template
MAIN_TLS_ENABLE = true
tls_on_connect_ports=465
```

Optionen

In der Datei */etc/default/exim4* muss noch die Variable SMTPLISTENEROPTIONS umgestellt werden. Sie werden Sie leer vorfinden. Ändern Sie sie bitte auf folgenden Wert:

```
# /etc/default/exim4
SMTPLISTENEROPTIONS='-oX 465:25 -oP /var/run/exim4/exim.pid'
```

Portbehandlung

Das Einschalten des TLS bewirkt nicht zwangsläufig, dass ein anderer Port als der Standardport 25 für SMTP verwendet wird. Viele Mailclients erwarten aber den SMTP-Server mit TLS auf Port 465. Die Option weist Exim an, beide Ports zu bearbeiten.

Damit sind die Konfigurationen für die Aktivierung des TLS abgeschlossen. Es muss nun das Zertifikat erstellt werden, das als Basis der Verschlüsselung verwendet wird.

Zertifikat

Selbstgebackenes

Sofern Sie nicht bereits ein Zertifikat besitzen, können Sie sich eines mit dem Skript *exim-gencert* erzeugen, das Sie in der Dokumentation von Exim finden. Dies erspart Ihnen einige etwas kryptische Aufrufe des Programms `openssl`. Sie rufen das Skript folgendermaßen auf:

```
debian # cd /etc/exim4
debian # /usr/share/doc/exim4-base/examples/exim-gencert
[*] Creating a self signed SSL certificate for Exim!
...
Generating a 1024 bit RSA private key
...
writing new private key to '/etc/exim4/exim.key'
...
Country Code (2 letters) [US]:DE
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company; recommended) []:
Organizational Unit Name (eg, section) []:
Server name (eg. ssl.dom.tld; required!):debian.willemer.edu
Email Address []:
[*] Done generating self signed certificates for exim!
...
debian #
```

Dateien

Es wurden die Dateien *exim.key* und *exim.crt* angelegt. Diese stellen Ihr persönliches Zertifikat dar. Damit das Zertifikat zur Kenntnis genommen wird, muss der Server neu gestartet werden.

```
debian # /etc/init.d/exim4 restart
```

Nicht autorisiert

Da das Zertifikat nicht von autorisierter Stelle signiert ist, wird ein E-Mailclient bei der ersten Verbindung husten und melden, dass hier ein großes Sicherheitsrisiko vorliegt. Setzen Sie Ihr James-Bond-Lächeln auf, und klicken Sie an, dass Sie sich des Risikos bewusst sind und dass Sie hiermit eine Ausnahme definieren. Danach wird er so lange Ruhe geben, solange sich das Zertifikat nicht ändert. Ein Man-In-The-Middle-Angriff ist damit abgewehrt, solange man beim ersten Kontakt sicher sein kann, dass der Mailserver der ist, den man erwartet.

22.4.6 Wer ist denn da?

Historische Altlast

In den Anfängen des Internets war die Anzahl der Benutzer noch überschaubar, und E-Mails wurden von Universitäten verwendet, um Forschungsergebnisse auszutauschen. Damit der Datenfluss funktionierte, schob jeder Mailserver gern für jeden anderen die Nachrichten weiter. Eine Hand wäscht die andere. Inzwischen besteht ein Großteil der Forschungen im Internet darin, den besten Preis für Komsumartikel zu finden oder nach einfältigen Menschen zu suchen, die ernsthaft glauben, dass in Nigeria mehrere Millionen Dollar verzweifelt auf einen neuen Besitzer warten.

Blacklist vereiteln

Um die Viagraverkäufer und Nigeria-Connections davon abzuhalten, den Mailserver mit Mails zu überfluten, mussten die Relays gegen den freien Zugang mit einem Passwort gesichert werden. Falls Ihr Server im Internet steht, gehört die Authentifizierung der Benutzer sogar zum Pflichtprogramm. Ein Server mit einem offenen Relay wird über kurz oder lang in den schwarzen Listen der Spambekämpfer landen. Und dann werden die Mails, die dieser Server verteilt, automatisch bei den meisten Empfängern in den Spamfilter wandern und ihr Ziel nie erreichen.

Ausblick

Im Folgenden werden drei Methoden vorgestellt, die Benutzerkontrolle durchzuführen. Zunächst gibt es die Möglichkeit, dass Exim eine eigene kleine Passwortdatei pflegt. So braucht nicht jeder Mail-User auch automatisch ein Konto auf dem Server selbst. Wenn sowieso nur die angemeldeten Nutzer des Servers den Mailserver nutzen sollen, liegt es nahe, die systemeigene Benutzerverwaltung zu verwenden. So bleiben die Passwörter immer gleich. Der dritte Weg, der hier aufgezeigt wird, ist die Verwendung einer Datenbank. Damit sollten auch größere Benutzeranstürme zu bewältigen sein.

Exims eigene Passwortdatei

Die einfachste Art, die Benutzer zu verwalten, ist die Einrichtung einer eigenen Passwortdatei. In der Datei */etc/exim4/exim4.conf.template* finden Sie folgende Zeilen hinter dem Label plain_server, die allerdings bei Ihnen noch mit einem Kommentarzeichen ausgeblendet sein dürften:

```
plain_server:
  driver = plaintext
  public_name = PLAIN
  server_condition = "${ifcrypteq{$3}{${extract{1}{:} \
      {${lookup{$2}lsearch{CONFDIR/passwd}{$value}{*:*}}}}} \
      {1}{0}}"
  server_set_id = $2
  server_prompts = :
  .ifndef AUTH_SERVER_ALLOW_NOTLS_PASSWORDS
  server_advertise_condition = ${if eq{$tls_cipher}{}{}{*}}
  .endif
```

Die Zeile server_condition wurde aus satztechnischen Gründen zweimal umgebrochen. Obwohl sie sehr kryptisch aussieht, können Sie durchaus erkennen, dass damit in der Datei *passwd* gesucht wird, die im Exim-Konfigurationsverzeichnis liegt. Diese Datei wird die eigene Passwortdatei für Exim sein. Nach einem Neustart von Exim werden die Benutzer bereits aus der Datei verwendet.

Dazu müssen natürlich erst einmal Benutzer in der neuen Passwortdatei vorhanden sein. Zur Pflege bietet sich der Befehl `htpasswd` an, den Sie vielleicht bereits vom Webserver Apache her kennen. Sie müssen für jeden Benutzer einen Eintrag anlegen. Als Parameter geben Sie vor dem Benutzernamen die verwendete Passwortdatei an.

htpasswd

```
debian # htpasswd /etc/exim4/passwd johannes
```

Mit der Option `-D` können Sie einen Benutzer aus der Passwortdatei wieder entfernen.

Lokale Benutzerverwaltung

Synchrone Passwörter

Sie können die Benutzerverwaltung auch an die lokale Benutzerverwaltung anschließen. Das ist eigentlich nur dann sinnvoll, wenn es wirklich darum geht, dem lokalen Benutzer einen eigenen SMTP-Server zur Verfügung zu stellen. Der Vorteil dieser Lösung ist, dass die Benutzer nicht unterschiedliche Passwörter für Mail- und Serverbetrieb benötigen. Die Übersättigung der Benutzer mit Passwörtern führt letztlich dazu, dass die Passwörter immer simpler ausfallen oder sogar aufgeschrieben werden. Beides kann nicht das Ziel eines sicherheitsbewussten Administrators sein.

SASL-Dämon

Für den Zugriff auf die lokale Authentifizierung verwenden wir den Dämon `saslauthd`, der mit dem Paket *sasl2-bin* installiert wird.

```
debian # apt-get install sasl2-bin
```

Dämonstart

In der Datei */etc/default/saslauthd* muss die Variable START auf »yes« gestellt werden, bevor der Dämon gestartet werden kann.

```
START=yes
```

Gruppenanschluss

Damit Exim mit dem Authentifizierungsdämon kommunizieren kann, muss er zur Benutzergruppe *sasl* gehören. Der folgende Befehl sorgt dafür:

```
debian # adduser Debian-exim sasl
```

Exim-Konfiguration

Nun wenden wir uns der Konfiguration von Exim zu. In der Datei *exim4.conf.template* finden Sie, hinter Kommentarzeichen verborgen, bereits alles, was Sie brauchen. Suchen Sie nach der Einstellung plain_saslauthd_server. Befreien Sie die zugehörigen Zeilen von den Kommentarzeichen!

```
plain_saslauthd_server:
  driver = plaintext
  public_name = PLAIN
  server_condition = ${if saslauthd{{$auth2}{$auth3}}{1}{0}}
```

```
  server_set_id = $auth2
  server_prompts = :
  .ifndef AUTH_SERVER_ALLOW_NOTLS_PASSWORDS
  server_advertise_condition = ${if eq{$tls_cipher}{}{}{*}}
  .endif
```

Start! Nun können Sie den Authentifizierungsserver starten und anschließend Exim neu starten, und die Anmeldung sollte funktionieren.

```
debian # /etc/init.d/saslauthd start
debian # /etc/init.d/exim4 restart
```

Benutzer in der Datenbank

Gemeinsame Datenbank Ein Mailserver, der seine Benutzer in einer Datenbank ablegt, wird dies nicht nur für fünf Benutzer tun. Es kann davon ausgegangen werden, dass es sich dabei schon um eine etwas umfangreichere Mailumgebung handeln wird. Und in diesem Fall wird es sicher auch einen POP3- oder IMAP-Server geben, der ebenfalls seine Benutzer verwaltet sehen möchte.

Courier-Paket Wenn Sie die Courier-Pakete verwenden, finden Sie bereits eine Konfiguration in der Datei *exim4.conf.template* hinter dem Label plain_courier_authdaemon vor. Tatsächlich verfügt das Courier-Paket über einen eigenen Authentifizierungsdämon, der ähnlich ansprechbar ist wie der SASL-Dämon.

Da die Authentifizierung über diesen Dämon läuft, müssen Sie nur das Courier-Paket konfigurieren und können Exim einfach daran anhängen. Sie finden dies im Workshop in Abschnitt 35.3 ab Seite 860 beschrieben.

Aufwand Ganz so billig, wie es sich hier anhört, ist die Einrichtung natürlich nicht. Sie müssen eine Datenbank anlegen, eine Tabellenstruktur definieren. Die Benutzerdaten müssen in diese Datenbank geschafft werden. Da nicht jeder Administrator davon begeistert sein wird, dies per SQL-Befehle einzutippen, werden Sie Skripte benötigen. Eine der kleinen Gemeinheiten ist das Verwalten der verschlüsselten Passwörter in der Datenbank. Dies finden Sie im Workshop in Abschnitt 35.3 ab Seite 860 anhand einer PostgreSQL-Datenbank beschrieben, funktioniert aber mit einer MySQL-Datenbank genauso gut.

22.4.7 Direktaufruf von Exim

exim4 Sie können mit dem Befehl `exim4` den MTA von außen befragen und ansteuern. Damit die Liste der Optionen im alltäglichen Gebrauch nicht

allzu lang wird, gibt es einige Aliasnamen des Befehls, die automatisch bestimmte Optionen setzen.

Warteschlange

So bewirkt der Befehl `mailq` dasselbe wie der Aufruf `exim4 -bp`: Er zeigt an, welche Nachrichten noch in der Warteschlange stehen.

Adresstest

Mit der Option `-bt` können Sie die Auslieferbarkeit einer E-Mail-Adresse prüfen. Die Ausgaben können Sie mit der Option `-d` für Debug noch erweitern. Als Parameter geben Sie die gesuchte Adresse an.

```
debian # exim -bt -d 9 peter@debian.willemer.edu
```

Option	Bedeutung
`-bp`	Zeigt die Mails in der Warteschlange
`-bpc`	Liefert die Anzahl der Mails in der Warteschlange
`-bt`	Testet eine Mail-Adresse
`-q`	Mails abarbeiten und versenden
`-Mrm` *Mail-ID*	Löscht die Mail *Mail-ID*

Tabelle 22.1 Optionen von exim4

22.5 Der Kampf gegen das Böse

E-Mail wäre so ein schönes Medium, wenn da nicht diese bösen Menschen wären, die fragen, ob Sie Viagra brauchen, auf zweifelhaftes Glücksspiel stehen, mit unechten Rolex-Uhren angeben wollen oder sich für rechtsradikales Gedankengut erwärmen möchten.

22.5.1 Spamassassin gegen Werbung

Installation

Debian verfügt über einen Spamfilter namens Spamassassin, der über das Paket gleichen Namens installiert wird.

```
debian # apt-get install spamassassin
```

Dämonbetrieb

Spamassassin läuft als eigenständiger Dämon im Hintergrund. Der Austausch mit dem MTA läuft über eine Socketverbindung. Um Spamassassin im Hintergrundbetrieb zu betreiben, müssen Sie in der Datei */etc/default/spamassassin* die Variable ENABLED auf 1 stellen. Das führt dazu, dass beim Systemstart Spamassassin automatisch gestartet wird.

```
# /etc/default/spamassassin
ENABLED=1
```

Konfiguration

Die Konfiguration findet in der Datei */etc/spamassassin/local.cf* statt. Dort stehen folgende Werte:

```
required_score 5  # Spam-Pegelwert
report_safe 1     # Bericht direkt in der Mail
bayes_auto_learn 1 # Spamassasin lernt selbst
```

Danach muss der Spamassassin noch einmal neu gestartet werden. Das Init-Skript tut dies.

```
debian # /etc/init.d/spamassassin restart
```

Exim vorbereiten

Damit Exim eine freundliche Verbindung zu Spamassassin aufbaut, sollte es das Schwergewicht aus dem Paket *exim4-daemon-heavy* sein. Hier gibt es bereits eine Schnittstelle zu Spamassassin, die Sie in der Datei */etc/exim4/exim4.conf.template* aktivieren. Dort kann die Adresse des Spamservers angegeben werden. In unserem Beispiel ist diese lokal. Spamassassin verwendet den Port 783.

```
spamd_address = 127.0.0.1 783
```

Im ACL-Bereich muss dann noch der Zugriff auf die lokalen Bereiche freigegeben werden. Dankenswerterweise sind die passenden Einträge bereits vorhanden und müssen nur noch von ihren Kommentarzeichen befreit werden.

```
acl_check_data:
   ...
  warn
     spam = Debian-exim:true
     message = X-Spam_score: $spam_score\n\
               X-Spam_score_int: $spam_score_int\n\
               X-Spam_bar: $spam_bar\n\
               X-Spam_report: $spam_report
```

Wenn Sie diese Zeilen entkommentiert haben, ist es an der Zeit, auch Exim neu zu starten.

```
debian # /etc/init.d/exim4 restart
```

Nun sollte alles bereit sein. Zum Testen benötigen Sie Spam. In der Datei */usr/share/doc/spamassassin/examples/sample-spam.txt* finden Sie eine entsprechende Beispielmail. Sie enthält die folgende Zeichenkette:

```
XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-
EMAIL*C.34X
```

Aus satztechnischen Gründen ist diese umgebrochen. Bitte tun Sie das in Ihrer Testmail nicht. Diese Zeile muss ohne Zeilenumbruch und ohne Leerzeichen gesendet werden. Spamassassin sollte sie erkennen und sofort als Spam identifizieren.

22.5.2 Virenschutz

Mail-Virenschutz

Insbesondere, wenn Sie Windows-Rechner im Netzwerk haben, kann es sinnvoll sein, die Mails nach Viren zu durchsuchen, bevor sie auf den Arbeitsplatzrechner gelangen. Da aber inzwischen die meisten Viren andere Wege als E-Mail-Anhänge verwenden, können Sie sich trotz Virenschutz auf dem Mailserver die Installation von Anti-Viren-Software auf den Arbeitsplätzen nicht ersparen. Sie benötigen dazu das Paket *clamav-daemon*.

```
debian # apt-get install clamav-daemon
```

Konfiguration clamd.conf

In der Konfigurationsdatei *clamd.conf* muss die Variable AllowSupplementaryGroups gesetzt sein.

```
# /etc/clamav/clamd.conf
AllowSupplementaryGroups true
```

Das ist sie in der Regel auch, sodass hier nichts zu tun ist. Nach dieser Meisterleistung muss der Dämon neu gestartet werden.

```
debian # /etc/init.d/clamav-daemon restart
```

Exim-Konfiguration

Wir wenden uns Exim zu. In der Datei */etc/exim4/exim4.conf.template* suchen Sie nach der Variablen av_scanner. Diese ist das Interface zu ClamAV. Dort befindet sich der Socket, über den Exim mit ClamAV kommuniziert.

```
# /etc/exim4/exim4.conf.template
# av_scanner = clamd:/tmp/clamd
av_scanner = clamd:/var/run/clamav/clamd.ctl
```

Sie suchen weiter nach dem Stichwort malware. Sie werden eine auskommentierte Zeilenfolge finden, die mit dem Wort deny beginnt. Dies ändern Sie.

```
# /etc/exim4/exim4.conf.template
  # deny
  #   malware = *
```

```
  #   message = This message was detected as possible malwa-
re ($malware_name).
  deny
    message = This message was detected as possible malwa-
re ($malware_name).
    demime = *
    malware = *
```

Neustart

Sie ahnen schon, dass es nun an der Zeit ist, Exim neu zu starten. Und wieder wird dazu das Init-Skript bemüht.

```
debian # /etc/init.d/exim4 restart
```

Aktualisierung

Die Virenkennungen werden durch den Dämon `freshclam` aktualisiert. Sie können dir relevanten Einstellungen durch den Befehl `dpkg-reconfigure` wiederholen.

```
debian # dpkg-reconfigure freshclam
```

Nachkonfiguration

Sie werden gefragt, ob Sie eine ständige Verbindung zum Internet haben und in welchem Takt Sie die Kennungen aktualisieren möchten.

22.6 POP3 und IMAP-Server Courier

Courier ist ein Paket aus mehreren Mailmodulen, die einzeln eingesetzt werden können. Vor allem der POP3- und der IMAP-Dämon haben sich weitgehend als Standard durchgesetzt. Der MTA, den Courier auch bietet steht in Konkurrenz zu Postfix und zu Exim, dem Standardmailserver unter Debian.

22.6.1 POP3-Server

Holt Mails aus Maildir

Ein POP3-Server liefert den Zugriff auf die lokalen Mails durch einen Benutzer, der sich mit Benutzername und Kennwort ausweisen muss. Der Courier-Server setzt wie die meisten POP3-Server nach der Installation auf der normalen UNIX-Mailumgebung auf. Er verwendet zur Authentifizierung die Passwortdatei des Systems und greift auf die Maildir-Struktur des MTA zu, um dem Client seine Mails zu liefern. Sie installieren die Courier POP3-Server mit dem folgenden Befehl:

```
debian # apt-get install courier-pop
```

Out of the box

Nach der Installation werden Anfragen von außen sofort beantwortet. Die Authentifizierung gelingt durch das lokale System. Allerdings arbeitet Courier grundsätzlich auf der Maildir-Struktur.

Maildir

maildirmake

Um eine korrekte Maildir-Umgebung zu schaffen, enthält das Courier-Paket das Programm `maildirmake`, mit dessen Hilfe ein Maildir-Verzeichnis und dessen Unterverzeichnisse angelegt werden. Ein Benutzer kann sein eigenes Maildir-Verzeichnis mit dem folgenden Befehl anlegen:

```
debian $ maildirmake $HOME/Maildir
```

TLS und Zertifikate

TLS-Variante

Der Zugriff auf POP3 sollte in allen öffentlichen Umgebungen verschlüsselt erfolgen. Dazu wird TLS (Transport Layer Security) verwendet. Um den TLS-fähigen POP3-Server zu installieren, verwenden Sie das Paket *courier-pop-ssl*.

```
debian # apt-get install courier-pop-ssl
```

Während der Installation wird automatisch ein selbst signiertes Zertifikat erzeugt, das in der Datei */etc/courier/pop3d.pem* abgelegt wird. Sie werden darauf hingewiesen, dass Sie, wenn Sie ein autorisiertes Zertifikat besitzen, dieses dort ablegen können.

Die Konfiguration der Zertifikate und der TSL-Umgebung erfolgt in den Dateien *pop3d.cnf* und *pop3d-ssl*. Beide finden Sie im Verzeichnis */etc/courier*.

22.6.2 IMAP-Server

Arbeitet auf dem Server

Im Gegensatz zu POP3, mit dessen Hilfe Sie Ihre Mails vom Mailserver auf Ihren Arbeitsplatzrechner kopieren, arbeiten Sie mit IMAP direkt auf dem Datenbestand Ihres Mailservers. Das IMAP-Paket des Courier-Pakets heißt *courier-imap* und wird mit dem folgenden Aufruf installiert:

```
debian # apt-get install courier-imap
```

Maildir

IMAP-Server erwarten die Mails im Maildir-Format. Der Courier-IMAP-Server ergänzt allerdings die Maildir-Struktur um einige Dateien, die aber rückwärtskompatibel sind. Fremde Programme können zwar in der Regel mit den Erweiterungen nichts anfangen, stören sich aber auch nicht daran. Die Benutzerauthentifizierung erfolgt durch das Debian-System. So kann der IMAP-Server direkt nach der Installation eingesetzt werden.

SSL und die Zertifikate

Verschlüsselt

Um die Kommunikation mit dem IMAP-Server zu verschlüsseln, benötigen Sie die SSL-Variante des IMAP-Servers. Diese finden Sie im Paket *courier-imap-ssl*.

```
debian # apt-get install courier-imap-ssl
```

Zertifikat

Bei der Installation wird automatisch ein Zertifikat erstellt, das natürlich nicht von einer autorisierten Zertfizierungsstelle abgezeichnet ist. Dies wird ein E-Mail-Programm beim ersten Zugriff sogleich bemängeln. Sollten Sie die Warnung aber übergehen und das unsichere Zertifikat damit bestätigen, wird der E-Mailclient erst dann wieder rebellieren, wenn sich das Zertifikat ändern sollte. Ist also der erste Kontakt gesichert, kann ein späterer Mann-In-The-Middle-Angriff ausgeschlossen werden. Vor allem ist natürlich eine mit nicht-autorisiertem Zertifikat erstellte Verschlüsselung immer noch besser als gar keine. Der Server kann sofort über den Port 993 erreicht werden, wenn bereits die Maildir-Struktur eingerichtet ist.

22.6.3 Benutzerverwaltung

Mail- und System-User

Standardmäßig setzen die Courier-Pakete auf der Benutzerverwaltung des Betriebssystems auf. Damit hat jeder Mail-Anwender automatisch ein Konto beim Betriebssystem. Sie können verhindern, dass sich Mail-Benutzer am System einloggen, indem Sie die Login-Shell in der Datei */etc/passwd* entfernen.

authdaemonrc

Welche Benutzerverwaltung die Courier-Pakete verwenden, wird in der Datei *authdaemonrc* im Verzeichnis */etc/courier* definiert. In der Variablen authmodulelist wird hinterlegt, auf welchem Weg die Pakete die Authentizität der Benutzer prüfen. Nach der Installation finden Sie hier den Wert authpam.

```
# /etc/courier/authdaemonrc
authmodulelist="authpam"
```

Virtuelle Benutzer

Die Courier-Module können aber auch selbst die Benutzerverwaltung übernehmen. Die so angelegten Anwenderkonten werden als virtuelle Benutzer bezeichnet, da Sie nicht als User im lokalen Betriebssystem existieren.

Courier kann die Benutzer in Datenbanken oder im LDAP ablegen. Beispielsweise können durch folgenden Eintrag festgelegt werden, dass die Benutzerdaten in einer Berkley DB abgelegt werden.

```
# /etc/courier/authdaemonrc
authmodulelist="authuserdb"
```

Speichermodule

Mit dem Eintrag dort ist es allerdings nicht immer getan. Sie benötigen ein Modul, das die Verbindung zwischen dem Mailserver und der Ablage schafft. Das Paket courier-authlib-userdb wird bereits bei jedem Courier-Paket mitgeliefert und ist in der Lage, Daten in einer Berkley DB abzulegen. Wollen Sie auf LDAP oder Datenbanken zugreifen, werden Sie noch weitere Pakete installieren müssen.

Konfigurationsdateien

Je nach Verfahren der Datenablage müssen Sie festlegen, welche Datenfelder des Speichermoduls welchem Feld des Mailservers entsprechen. Bei Verwendung der Passwortdatei ist der Zusammenhang eindeutig, bei einer MySQL-Datenbank müssen die Feldnamen beschrieben werden. Diese Aufgabe über nimmt jeweils eine rc-Datei, die je nach verwendetem Authentifizierungsmechanismus unterschiedlich heißt.

In Tabelle 22.2 werden der Wert der Variablen authmodulelist, das benötigte Paket und die Konfigurationsdatei gegenübergestellt.

authmodulelist	Debian-Paket	Konfigurationsdatei
authpam	–	–
authuserdb	courier-authlib-userdb	–
authpgsql	courier-authlib-postgresql	authpgsqlrc
authldap	courier-authlib-ldap	authldaprc
authmysql	courier-authlib-mysql	authmysqlrc
authpipe	courier-authlib-pipe	–

Tabelle 22.2 Verwaltung der Benutzer

Benutzer vmail

Soll die Benutzerverwaltung von der Passwortdatei des Systems abgekoppelt werden, müssen die Arbeiten durch einen eigenen Benutzer des Mailsystems durchgeführt werden. Der Name ist frei wählbar. Wir verwenden hier *vmail*. Er muss im System als Benutzer angemeldet werden. Für diesen Benutzer *vmail* muss noch ein Benutzerverzeichnis angelegt werden und darin das Verzeichnis für die Mail.

```
debian # useradd vmail
debian # mkdir /home/vmail
debian # chown vmail /home/vmail
debian # su vmail
debian $ mkdir /home/vmail/domain
debian $ mkdir /home/vmail/domain/user
debian $ maildirmake /home/vmail/domain/user/Maildir
```

maildirmake Der Befehl `maildirmake` aus dem Courier-Werkzeugkasten erzeugt eine Maildir-Struktur. Er legt nicht nur den Ordner an, sondern baut alle benötigten Unterstrukturen auf.

Workshop Sie finden ein Beispiel für eine IMAP-Installation auf der Basis einer PostgreSQL-Datenbank im Workshop in Abschnitt 35.3 ab Seite 860.

22.7 Post sammeln: Fetchmail

Wenn Ihr Mailserver nicht mit einem MX-Eintrag in dem für Ihre Domain zuständigen DNS-Server[4] hinterlegt ist, ist es nicht ganz einfach, an Ihre Post zu kommen. Und selbst wenn Ihr Server als offizieller Domain-Mailserver eingetragen ist, werden Sie Probleme mit Mail-Adressen fremder Provider haben, die keine Möglichkeit der Weiterleitung von Mails anbieten.

Lückenbüßer Fetchmail schließt diese Lücke. Das Programm meldet sich wie ein normaler Mailclient beim Provider an, holt die Post ab, sortiert sie nach Benutzern und fügt sie per SMTP in den Briefschlitz des lokalen Mailservers.

Installation und Start

Installation Fetchmail kommt in einem eigenen Paket namens *fetchmail* und wird mit dem folgenden Befehl installiert:

```
debian # apt-get install fetchmail
```

Programmstart Das Programm kann direkt gestartet oder als Dämon betrieben werden. In der Regel wird der Hintergrundbetrieb interessanter sein. Soll allerdings Fetchmail beim Aufbau einer Internetverbindung aktiviert werden, müsste es aus einem PPP-Skript gestartet werden. Und dazu ist der direkte Aufruf nützlich. Auch in der Konfigurationsphase ist der gezielte Aufruf vorteilhaft. Sie könnten den Direktaufruf auch für die `crontab` (siehe Abschnitt 11.3 Seite 366) verwenden. Wollen Sie Fetchmail allerdings als Dämon laufen lassen, editieren Sie die Datei */etc/default/fetchmail*. Dort finden Sie die folgende Zeile:

```
# /etc/default/fetchmail
START_DAEMON=no
```

Dämonisierung Wenn Sie den Wert auf »yes« ändern, wird Fetchmail im Hintergrund laufen. Alternativ können Sie beim Direktaufruf auch die Option `-d` ver-

4 DNS siehe Kapitel 21 Seite 651

wenden. Dahinter geben Sie die Anzahl der Sekunden an, die zwischen zwei Aufrufen vergehen sollen.

Postfächerliste fetchmailrc

Konfiguration durch .fetchmailrc

Konfiguriert wird das Programm über die Datei *.fetchmailrc* des jeweiligen Benutzers. Sie muss im jeweiligen Benutzerverzeichnis des Aufrufers stehen. Eine Zeile in dieser Datei hat folgenden Aufbau:

Struktur einer Zeile in der Datei .fetchmailrc

`poll` *<Server>* `protocol POP3 user` *<User>* `password` *<passwd>*
`is` *<localUser>* [`keep`]

Passwortschutz

Leider muss hier das Passwort im Klartext stehen. Damit nicht jeder die Passwörter auslesen kann, muss die Datei mit `chmod 600 .fetchmailrc` für Fremde unlesbar gemacht werden. Ist das nicht der Fall, verweigert Fetchmail seinen Dienst und weist auf das Problem hin.

Mail nicht löschen

Die Option `keep` am Ende der Zeile sorgt dafür, dass die Mail auf dem Server nicht gelöscht wird. Wird beim Direktaufruf die Option `-k` verwendet, löscht Fetchmail in keinem Briefkasten die gelesenen Mails.

Alle Mails holen

Fetchmail liest nur bisher ungelesene Mails. Sollen alle gelesen werden, verwenden Sie beim Aufruf die Option `-a`.

Protokoll

Die Option `-L` schreibt die Protokolle in die Datei, deren Name als Parameter folgt.

Option	Bedeutung
`-k`	Mails auf dem POP3-Server nicht löschen
`-a`	Liest alle Mails, nicht nur die bisher ungelesenen
`-L` *Datei*	Protokolliere in *Datei*

Tabelle 22.3 Optionen von fetchmail

22.8 Postfix, die weitverbreitete Alternative

Alternative zu sendmail

Wietse Venema begann das Projekt Postfix im Rahmen eines Forschungssemesters im T. J. Watson Research Center der Firma IBM. Zu jener Zeit herrschte das Programm `sendmail` als Standardmailsystem vor. Das Programm `sendmail` ist ungeheuer leistungsfähig, durch seine gewachsenen

Strukturen und die kryptischen Konfigurationsdateien aber nicht überall gleichermaßen beliebt.

Populär

Das Projekt Postfix hatte darum vor allem das Ziel, den bisher monolithischen Mail Transport Agent durch mehrere schlanke Programme zu ersetzen. Die Konfiguration sollte vereinfacht und die Sicherheit erhöht werden. Inzwischen ist Postfix einer der gängigsten Mailserver überhaupt. Aus diesem Grund setzen einige Programmpakete auf Postfix auf, wie beispielsweise die Groupware Kolab oder die ISP-Software ispCP.

22.8.1 Installation

Debian liefert standardmäßig Exim4. Wenn Sie einen anderen Mailserver installieren, wird Exim4 zwangsläufig deinstalliert. Sie ersetzen Exim4 durch Postfix mit dem folgenden Befehl:

```
debian # apt-get install postfix
```

Dialogkonfiguration

Nach dem Herunterladen der Pakete werden Sie aufgefordert, die Konfiguration von Postfix im Dialog zu betreiben. Wenn Sie genau wissen, was Ihr Postfix tun soll, kann Ihnen dieser Assistent etwas Arbeit abnehmen. Nach der Installation können Sie diese Art der Konfiguration mit dem Befehl `dpkg-reconfigure` jederzeit noch einmal ausführlich nachholen.

```
debian # dpkg-reconfigure postfix
```

Dateikonfiguration

An dieser Stelle wird allerdings beschrieben, wie die Konfiguration über die Konfigurationsdateien erfolgt. Also beenden Sie den Assistenten einfach, indem Sie KEINE KONFIGURATION auswählen. Dies empfiehlt sich auch, falls Sie bereits eine Konfiguration vorliegen haben sollten. Der Assistent zwingt in jedem Fall den Postfix MTA zum Schluss dazu, die Konfigurationsdateien noch einmal zu lesen.

main.cf

Die Eingaben des Assistenten zielen in erster Linie auf Änderungen in der Datei */etc/postfix/main.cf* ab. Wenn Sie diese Datei ändern, rufen Sie anschließend den Befehl `postfix reload` auf, um die Konfigurationsänderungen zu aktivieren.

22.8.2 Konfiguration

Im Verzeichnis */usr/share/postfix* finden Sie ein paar Beispiele für die Konfigurationsdateien. Insbesondere die Datei *main.cf.dist* ist hier sehr hilfreich, da sie die Konfigurationsparameter mit recht ausführlichen Kommentaren enthält.

main.cf

In der Konfigurationsdatei werden Parameter als Zuweisungen gesetzt. Zunächst erscheint der Name des Parameters, dann ein Gleichheitszeichen und schließlich der Inhalt. Der Inhalt kann in der nächsten Zeile fortgesetzt werden, wenn diese mindestens mit einem Leerzeichen beginnt. Daraus folgt natürlich auch, dass ein neuer Parametername immer ganz links beginnen muss.

Wer bin ich eigentlich?

Mit den Variablen `myhostname` und `mydomain` werden der Rechner und die Domäne festgelegt, die von anderen Parametern verwendet werden. Dazu gehört insbesondere die Variable `mydestination`. In normalen Konfigurationen steht hier der Name des lokalen Rechners und der lokalen Domäne. Werden die beiden Variablen nicht explizit in der Datei *main.cf* gesetzt, enthält die Variable `myhostname` den Wert, den der Systemaufruf `hostname()` liefert. Der Domänenanteil des Aufrufs wird standardmäßig als Wert für die Variable `mydomain` verwendet.

Lokale Mail

Die Variable `mydestination` nimmt die Domäne auf, für die die Mail lokal verteilt wird. Hier sollte in meinem Fall also nicht *willemer.de* stehen, da mein Bruder beispielsweise auch eine Adresse dieser Domäne hat und in diesem Fall eine Mail an ihn mein privates Hausnetz nie verlassen würde. Aus diesem Grund habe ich für das lokale Netzwerk die Domäne *willemer.edu* erfunden, in der Annahme, dass wohl niemand eine amerikanische Bildungsstätte nach mir benennen wird.

Es wird in der Dokumentation von Postfix vorgeschlagen, für `mydestination` folgenden Eintrag zu wählen:

```
mydestination = $mydomain $myhostname localhost.$mydomain
```

Auf einem Mailserver für eine Domäne gibt es auch keinen Grund, diesen Eintrag zu ändern. Wenn ein anderer Rechner in meinem Netzwerk auch E-Mails versenden will, muss er seine Mails an *squeeze* weiterleiten, weil dieser den Austausch mit dem Internet durchführt. Ein solcher Rechner würde also gar keine Mails lokal abwickeln, sondern alle Mails an den zentralen Rechner schicken. Dieser Rechner hätte also folgende Einstellung:

```
mydestination =
relayhost = squeeze
```

Alle Mails würden dadurch an den Rechner *squeeze* weitergeleitet, der seinerseits prüft, ob es sich um lokale Mails handelt oder ob sie ins Internet weitergeleitet werden müssen.

Wählverbindung mit dem Internet

relayhost

Ist der Rechner über eine Wählleitung mit dem Provider verbunden, soll Postfix die Post bei geschlossener Verbindung speichern und bei offener Verbindung an den Rechner weitergeben, den der Provider zur Verfügung gestellt hat. Den Namen dieses Rechners erfahren Sie bei Ihrem Provider. Bei T-Online heißt er beispielsweise *mailto.t-online.de* und bei 1&1 *smtp.1und1.de*. Ein solcher Rechner wird als Relay bezeichnet und in der Konfigurationsdatei *main.cf* unter dem Namen `relayhost` festgelegt.

```
relayhost = smtp.1und1.de
```

defer_transports

Damit die Mail nicht sofort verteilt wird, sondern erst bei einer Verbindung mit dem Internet, wird der Parameter `defer_transports` auf `smtp` gesetzt:

```
defer_transports = smtp
```

Da der Server direkt angegeben wird, braucht nicht per DNS der Mailserver der Domäne über den MX-Eintrag gesucht werden. Also schalten Sie das Suchen über DNS ab:

```
disable_dns_lookups = yes
```

Damit die Mail den Rechner in Richtung Internetprovider verlässt, muss das Kommando `sendmail -q` gesetzt werden. Dieser Aufruf heißt absichtlich genauso wie der Befehl, der bei einer `sendmail`-Installation verwendet wird, damit die Skripte nicht angepasst werden müssen.

22.8.3 Mbox und Maildir

Lokale Mbox

Nach der Grundkonfiguration verteilt der Postfix-Dämon die lokale Post. Dabei werden die Nachrichten im Mbox-Standard im Verzeichnis */var/mail* abgelegt. Für jeden Benutzer gibt es dort eine Datei, die all seine Post aufnimmt. Sie können dies ausprobieren, indem Sie mit dem Befehl `mail` Post versenden.

SMTP vom Netzwerk

Auch der Zugriff von einem anderen Rechner des gleichen Netzwerks funktioniert ohne vorherige Konfiguration. Sie können dies ausprobieren, indem Sie den Befehl `telnet` aufrufen, als ersten Parameter den Mailserver und als zweiten Parameter die Portnummer 25 für SMTP angeben.

```
squeeze $ telnet debian 25
Trying 192.168.109.199...
Connected to debian.willemer.edu.
Escape character is '^]'.
```

```
220 debian.willemer.edu ESMTP Postfix (Debian/GNU)
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

Die Verbindung wird aufgebaut, und Postfix meldet sich artig. Die Eingabe des Wortes »QUIT« löst die Verbindung, Postfix verabschiedet sich mit dem Wort »Bye«.

Bediente Interfaces

Sie wollen nicht, dass Postfix alle Schnittstellen bedient? Dann sollten Sie in der Datei *main.cf* den Wert der Variablen `inet_interfaces` vom Wert »all« auf die IP-Adressen der Schnittstellen beschränken, die in Zukunft bedient werden sollen. Bei mehreren Adressen trennen Sie diese durch ein Leerzeichen. Die folgende Definition erlaubt den localhost und das lokale Netzwerk:

```
inet_interfaces = 127.0.0.1 192.168.109.199
```

Umstieg auf Maildir

In der Konfigurationsdatei *main.cf* im Verzeichnis */etc/postfix* wird die Variable `home_mailbox` auf den Wert `Maildir/` gesetzt. Anschließend muss Postfix die Datei neu einlesen. Der Befehl `postfix reload` erreicht dies. Die Konfiguration kann auch durch den Befehl `postconf` geändert werden.

```
debian # postconf -e "home_mailbox=Maildir/"
debian # postfix reload
```

Anschließend können Sie einen POP3- oder einen IMAP-Server installieren, der dann direkt auf die Maildir-Struktur aufsetzt.

22.8.4 Lookup-Tabellen

In der Datei *main.cf* ist unter dem Stichwort »maps« eine Reihe von Lookup-Tabellen angegeben. Diese liegen als ASCII-Textdateien vor, müssen aber, bevor Postfix sie lesen kann, mit dem Programm `postmap` in je eine Datenbankdatei gewandelt werden. Ein typisches Beispiel dafür ist die Datei *canonical*. Sie enthält eine Tabelle, die Benutzernamen des Systems in die Namen umsetzt, die vor dem @-Zeichen stehen. Beispielsweise könnte der Benuter *till* ein E-Mail-Adresse till.eulenspiegel@debian.willemer.edu verwenden wollen.

```
# /etc/postfix/canonical
till till.eulenspiegel@debian.willemer.edu
```

Wenn Sie darin Änderungen vornehmen, müssen Sie die Datei mit dem Befehl `postmap` in eine db-Datei wandeln.

```
debian # cd /etc/postfix
debian # postmap canonical
```

Es entsteht eine Datei namens *canonical.db*. Der passende Eintrag in der Datei *main.cf* lautet:

```
# /etc/postfix/main.cf
canonical_maps = hash:/etc/postfix/canonical
```

Die Datei *canonical* bewirkt das »input address rewriting«. Das Gegenstück des »output address rewriting« wird in der Datei *generic* festgelegt. Für beide Dateien gibt es eine eigene Manpage.

22.8.5 Warteschlangen

Postfix arbeitet mit mehreren Warteschlangen, die sich unterhalb des Verzeichnisses */var/spool/postfix* befinden. Die wichtigsten dieser Warteschlangen sind:

- **incoming**
  Hier befindet sich die eintreffende Mail.
- **maildrop**
  Dies ist der Ort, an dem der Anwender seine E-Mail einwirft.
- **active**
  Hier liegt die Mail, die zur Weiterleitung freigegeben ist.
- **deferred**
  In diesem Verzeichnis wird Mail zurückgehalten, die noch auf ihre Weiterleitung warten muss.

Welche Nachrichten sich in welcher Warteschlange befinden, erfahren Sie über den Befehl `postqueue -p`. Sollen alle Mails, die beispielsweise durch eine temporäre Störung verzögert werden, wieder weitergeleitet werden, verwenden Sie den Befehl `postsuper -r ALL`.

22.8.6 Virtuelle Domänen

Natürlich bietet Postfix ebenfalls die Möglichkeiten, virtuelle Domains zu verwalten. Dieses Thema wurde am Beispiel von Exim4 bereits de-

taillierter vorgeführt. Aus diesem Grund soll dies nicht alles für Postfix noch einmal wiederholt werden. Sie finden darüber hinaus im Internet für Postfix zahllose Beispiele, da Postfix eben weit verbreitet ist. Hier wird also nur in groben Zügen umrissen, wie die Einrichtung bei Postfix durchgeführt wird.

main.cf

In der Datei *main.cf* verweisen einige Parameter auf die weiterführenden Konfigurationsdateien. Die folgenden Zeilen zeigen Beispiele für eine Konfiguration, die auf eine PostgreSQL-Datenbank zugreift. Statt »pgsql« können Sie »mysql« einsetzen, wenn Sie lieber eine MySQL-Datenbank verwenden wollen.

```
# /etc/postfix/main.cf
virtual_alias_maps = pgsql:/etc/postfix/virtual_alias
virtual_mailbox_domains = pgsql:/etc/postfix/virtual_domains
virtual_mailbox_maps = pgsql:/etc/postfix/virtual_mailboxes
virtual_mailbox_base = /var/mail/vmail
```

DB-Konfiguration

Die Konfigurationsdatei enthält den Benutzer, Passwort, Datenbanknamen und vor allem den `SELECT`-Befehl, mit dem die gewünschten Daten aus der Datenbank geholt werden. Hier als Beispiel ein Zugriff auf die Domaintabelle:

```
# /etc/postfix/virtual_domains
user = maildbuser
password = ganzgeheim
dbname = mail
query = SELECT domain FROM domains WHERE domain='%s'
hosts = 127.0.0.1
```

Datenbank anlegen

Natürlich muss in der Datenbank eine entsprechende Struktur angelegt worden sein. Wie im Workshop in Abschnitt 35.3 ab Seite 860 gezeigt, kann die Datenbank auch vom POP3- und den IMAP-Server genutzt werden, um dessen Zugriffe zu authentifizieren.

Index

B

C

D

E

F

J

K

L

M

N

O

P

Q

T

U

V

W

X

Y

Z